MSc thesis in Geomatics

Inferring the number of floors of building footprints in the Netherlands

Ellie Roy

January 2022

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Dr. Hugo Ledoux
	Dr. Giorgio Agugiaro
	Ir. Maarten Pronk
Co-reader:	Ir. Balázs Dukai

Abstract

Data on the number of floors is required for a variety of applications, ranging from energy demand estimation to flood response plans. Despite this, open data on the number of floors is currently not available at a nationwide level in the Netherlands. This means that it must be inferred from other available data. Automatic methods usually involve dividing the estimated height of a building by an assumed storey height. In some cases, this simple approach limits the accuracy of the results. Therefore, the goal of this thesis is to develop an alternative method to automatically infer the number of floors.

Three different machine learning algorithms are tested and compared: Random Forest, Gradient Boosting and Support Vector Regression. These algorithms are trained using data on the number of floors obtained from four municipalities in the Netherlands. In addition, 25 features are derived from cadastral attributes, building geometry and neighbourhood census data. These features are tested in different combinations in order to determine whether a specific subset yielded better results. Furthermore, a comparison is made between features derived from 3D building models at different levels of detail.

The results show that building height, particularly 70th percentile height, is most related to the number of floors. Other 3D geometric features are also found to be quite closely related to the number of floors, specifically roof area and volume. However, a higher level of detail did not improve the results. Cadastral features are also found to be relevant; mainly net internal area and, to a lesser extent, construction year. Furthermore, models based on a combination of different features performed better than models based on single categories of features.

The best predictive model achieved an accuracy of 94.5% and a Mean Absolute Error (MAE) of 0.06 for buildings with 5 floors or less. This represented a substantial improvement on the results of the geometric approach, which had an accuracy of 69.9% and MAE of 0.31. However, above 5 floors, model performance was substantially lower. Machine learning provided only a slight improvement on the geometric approach for these buildings. In this case, the best model had an accuracy of 52.3% and MAE of 0.62, whereas the geometric approach was 47.5% accurate and had a MAE of 0.70. A comparison of the cumulative error distributions showed that the best model mainly improved the fraction of buildings that were predicted with an error of less than 1 floor. Overall, these results show that machine learning partially provided a better estimate of the number of floors than a purely geometric approach.

Acknowledgements

Firstly, I would like to thank my main supervisor, Hugo Ledoux, for the valuable feedback and guidance he provided throughout the thesis. I would also like to thank my other supervisors, Giorgio Agugiaro and Maarten Pronk, for providing useful suggestions and insights. I am also grateful to Camilo León-Sánchez for helping to brainstorm ideas and assisting with some technical aspects of the implementation. Finally, I would like to thank Sia for his support and patience.

Contents

List of Figures

List of Tables

Acronyms

1 Introduction

Many analyses that are based on building datasets rely on the availability of specific attributes [Biljecki et al., 2015; Krayem et al., 2021]. In particular, data on the number of floors is required for a variety of applications. For instance, this information is used in the estimation of building energy demand, which allows the benefit of energy retrofitting to be assessed [Agugiaro, 2015; Nouvel et al., 2014]. It can also be used to estimate building population, which is useful for various network analysis and urban planning applications [Lwin and Murayama, 2009; Krayem et al., 2021].

Furthermore, data on the number of floors can be used to determine the amount of inhabitable storeys remaining during a flood (M. Pronk, personal communication, July 28th, 2020). This is a particularly relevant application for the Netherlands, as approximately half of the population live at flood risk. For this reason, the Dutch government has developed a website called www.overstroomik.nl, which provides information on whether any dry storeys would remain in each building given a major breach of the country's flood defences.

Despite the wide range of applications, open data on the number of floors is often unavailable [Biljecki and Sindram, 2017]. The most detailed, openly available dataset on buildings and addresses in the Netherlands is the *Basisregistratie Adressen en Gebouwen (BAG)*. Within this dataset, the footprints of all buildings are stored as 2D polygons. Each footprint is associated with a number of attributes, such as construction year and current use, but the number of floors is currently not included. In some cases, this attribute is maintained at a municipal-level in a more extensive version of the dataset called the BAG "plus" (BAG+). However, generally this data is only maintained for internal use and not made openly available [Heeres, 2016].

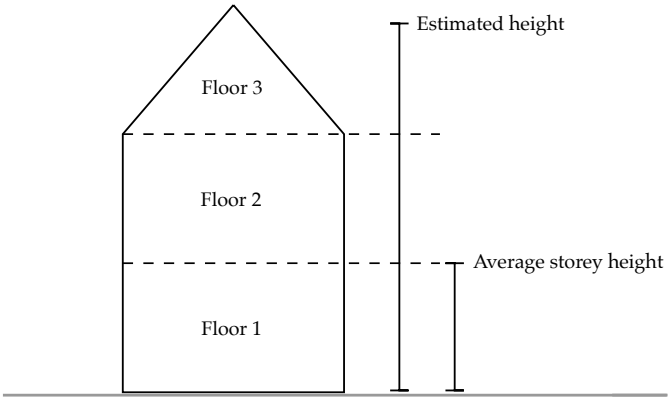


Figure 1.1: Geometric approach to estimate number of floors from building height

The lack of open data on the number of floors means that it must be inferred from other available data. Automatic methods are often based on building geometry, which usually involves dividing the estimated height of a building by an assumed storey height (Figure 1.1). Building height can be determined by combining the geometry of the 2D footprints with point cloud data (Figure 1.2). In the Netherlands, the point cloud from the *Actueel Hoogtebestand Nederland (AHN)* can be used for this purpose. This is a nationwide elevation model obtained through airborne Light Detection and Ranging (LiDAR). The 3D geoinformation group at TU Delft has automated the process of extracting building height from the AHN point cloud and uses this to provide a 3D version of the BAG at different levels of detail. These 3D building models are made openly available through their 3D BAG service [Dukai, 2018].

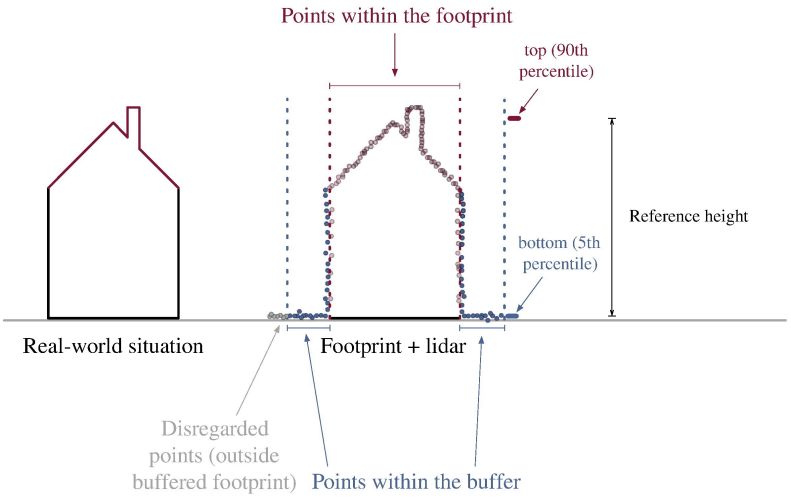


Figure 1.2: Determination of building height from aerial LiDAR data. Adapted from Biljecki et al. [2017].

Given the strong correlation between building height and floor count [Biljecki et al., 2017], using height to estimate the number of floors can often provide reliable results. However, in certain cases the over-simplification of building geometry limits the accuracy of the estimation, which can have adverse consequences for the intended application. This is further explained in Section 2.2.1.

1.1 Research objectives

Due to the limitations of the current approach, the goal of this thesis is to develop an alternative method to automatically infer the number of floors. This method will be based on the building footprints available in the BAG dataset. Similar studies related to inferring building properties have used machine learning to obtain accurate results. These studies show that combining multiple attributes has a greater potential than relying on a single predictor. For this reason, the alternative method will also focus on using machine learning to combine multiple building attributes, such as roof shape and construction year, in order to obtain a more realistic estimate of the number of floors.

Based on this objective, the main research question is formulated as follows:

To what extent can machine learning provide a better estimate of the number of floors than a purely geometric approach?

In order to achieve the main research objective, the following sub-questions are defined:

a. Which features are related to the number of floors? Is there any overlap between these features and which subset yields the best results?

b. Which machine learning algorithm provides the best results? How are the results affected by feature subsets that reflect different levels of data availability?

c. What level of performance can be achieved compared to a purely geometric approach? What types of gross errors are present?

d. Since floor count is generally an integer value, is this a regression or classification problem? If considered regression, how does rounding the predictions affect the results?

1.2 Scope

The following scope is defined for the research:

- The focus will be on buildings in the Netherlands due to:
  - the wide availability of open data
  - the aim to integrate the number of floors as a new attribute of the 3D BAG service
  - the fact that applications related to flooding are specifically relevant to the Netherlands
- The focus will be on (mixed-)residential buildings because:
  - this avoids over-complication of the problem, since non-residential buildings are generally more variable in design
  - some applications are most relevant to this type of building use (e.g the inhabitability of homes subject to flooding)
- The comparison of different machine learning algorithms will not be extended to deep learning, in order to prevent the algorithm from becoming a "black box"
- The methodology will not focus on modelling building geometry, as 3D building models are already openly available at a nationwide level through the 3D BAG

1.3 Outline

This thesis consists of five main chapters. Chapter 2 presents the background and previous work related to this thesis. Fundamental aspects of 3D building models are introduced, as well as the machine learning techniques used. Furthermore, geometric approaches to estimate the number of floors are explained in further detail, including a discussion of their limitations. Lastly, previous applications of machine learning to generate or enrich 3D building models are discussed.

Chapter 3 provides an overview of the methodology used to address the research objectives. Each step of the methodology is explained from a conceptual standpoint. Then, further details on the implementation are provided in Chapter 4. In Chapter 5, the results of the methodology are presented. The best predictive model is selected and analysed in further detail, including a comparison to the results obtained using a geometric approach.

Finally, Chapter 6 concludes the thesis. First, the research questions are answered, in order to determine the extent to which the objectives were fulfilled. This is followed by a discussion of the strong and weak aspects of the approach. Based on this discussion, recommendations for future work are provided. In addition, new research questions that emerged during the analysis are discussed.

2 Background and related work

This chapter presents the background and related work that this thesis builds upon. It focuses on four main topics: (1) fundamental aspects of 3D building models relevant to this research, (2) geometric approaches used to estimate the number of floors, (3) machine learning techniques and (4) previous applications of machine learning to generate or enrich 3D building models.

2.1 3D building models

2.1.1 Level of Detail

An important aspect of 3D building models is their Level of Detail (LOD). This concept describes the complexity of a model, allowing its degree of resemblance to the real-world situation to be portrayed [Biljecki et al., 2016b]. The most widely used standard for specifying the LOD is defined by the Open Geospatial Consortium (OGC) [OGC, 2012; Gröger and Plümer, 2012]. This standard consists of five distinct categories of increasing geometric and semantic complexity (Figure 2.1), defined as follows:

- LOD0: 2D building footprint
- LOD1: Block model, usually obtained by extruding the LOD0 model
- LOD2: General model of the building structure including simplified roof shapes and semantic classes to describe the boundary surfaces (e.g. wall, roof)
- LOD3: Architecturally detailed model including roof and wall structures (e.g. windows, doors)
- LOD4: Complete model of the building including its interior

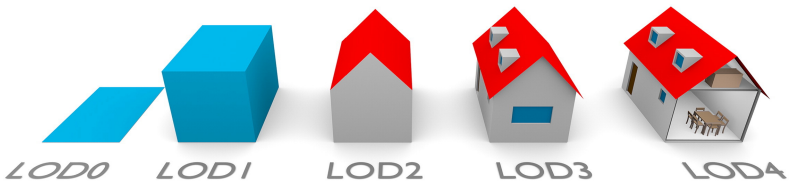


Figure 2.1: The five LODs defined by the OGC CityGML 2.0 standard.
Reprinted from Biljecki et al. [2016a].

This categorisation has been further refined by Biljecki et al. [2016a] to consist of four sub-categories per LOD, with the exclusion of LOD4 which is rarely used in practice. The refined LOD specification is shown in Figure 2.2. It reduces the ambiguity of the OGC categorisation by providing a more precise definition of the geometric detail required within each sub-category [Biljecki et al., 2016a].

In the Netherlands, LOD1.2, 1.3 and 2.2 models are readily available through the 3D BAG. These models can be used to extract geometric features potentially relevant to estimating the number of floors, such as volume and roof area. Part of this thesis focuses on determining whether features based on a higher level of detail provide better predictions. In order to reduce the complexity of the analysis, only the highest and lowest levels of detail (LOD1.2 and 2.2) were considered. LOD1.2 was also chosen rather than LOD1.3 because it is the most commonly used LOD1 model in practice [Biljecki et al., 2016a]. This is because it can be easily obtained by extruding each building footprint to a uniform height. In contrast, modelling the differentiated roof heights present in LOD1.3 requires more complex algorithms. Therefore, an analysis based on LOD1.2 was also more widely applicable.

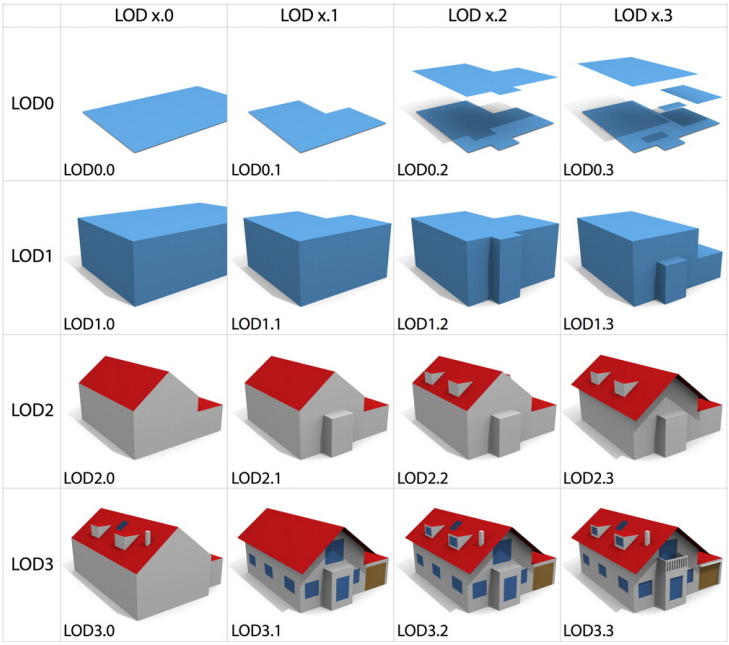
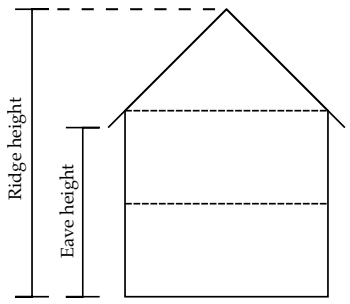


Figure 2.2: Improved LOD specification. Reprinted from Biljecki et al. [2016a].

2.1.2 Height references

Despite the apparent simplicity of the LOD1 block model, there is a high level of ambiguity in its geometric representation [Biljecki et al., 2014]. As illustrated in Figure 2.3a, the position of the top surface varies significantly depending on the geometric reference chosen to represent the building's height. In LOD1, these geometric references can be taken into account by calculating building height at different percentiles of the point cloud's z-coordinates [Dukai et al., 2019]. In the context of this thesis, the geometric references representing the ridge and eaves are the most important (Figure 2.3b). The difference between these heights could potentially be used to identify storeys beneath slanted roofs.

(a) Different height references for the top surface of an LOD1 model. Reprinted from Biljecki et al. [2014].



(b) Ridge vs. eave height

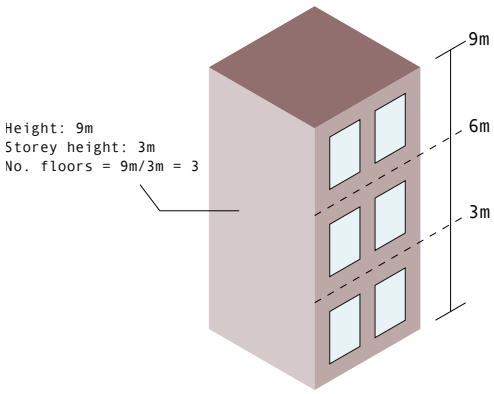
Figure 2.3: Geometric height references

2.2 Geometric approaches to estimate number of floors

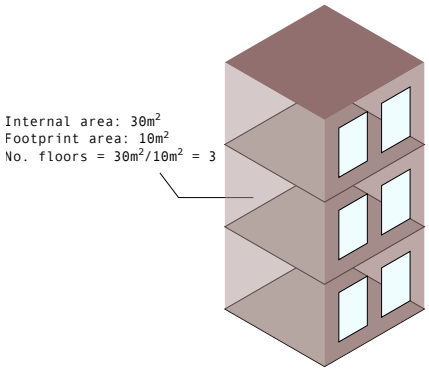
When information on the number of floors is unavailable, commonly used approaches to estimate this attribute are based on building geometry. These approaches are either height- or area-based. The following sub-sections describe these approaches in more detail and discuss their limitations.

2.2.1 Height-based approach

The height-based approach provides an estimate of the number of floors by dividing the height of a building with an assumed storey height (Figure 2.4a). The results are typically rounded to the nearest integer, but may also be rounded up or down depending on the application. This approach takes advantage of the strong correlation between building height and the number of floors (i.e. taller buildings have more storeys). It can work well for buildings with flat roofs, as the roof surface corresponds to a single height. However, for slanted roofs the results are highly dependent on the geometric reference chosen to represent building height. In the Netherlands only 34% of buildings are estimated to have a truly flat roof [Dukai et al., 2019], meaning that choosing an appropriate height reference is important. In addition, it is also important to select a representative storey height. This is difficult because the storey height of different buildings varies depending on a number of factors, such as building type and age. Some buildings may also have varying heights per storey (e.g. a ground floor lobby taller than the floors above). This variation in storey height means that, despite a strong correlation between the number of floors and building height, there is also substantial variation in the number of floors of buildings with the same height. However, residential buildings are generally observed to exhibit more consistent trends than non-residential buildings [Biljecki et al., 2017].



(a) Height-based



(b) Area-based

Figure 2.4: Geometric approaches to estimate number of floors

The height-based approach is cited by a number of research papers [Shiravi et al., 2015; Alahmadi et al., 2013; Nouvel et al., 2014]. It is also the main method used to estimate the number of floors for the flood response plans developed for the `www.overstroomik.nl` application (M. Pronk, personal communication, July 28th, 2020). In this case, roof height is calculated as the 75th percentile of each building's point cloud and a single storey height of 2.65 metres is used. This value was derived from the Dutch building code (*Bouwbesluit*) and is the average of the standards used before and after 2003. In order to provide a conservative estimate of the number of floors, the results are rounded down. This reduces the likelihood of overestimation, which could potentially cause the presence of dry floors to be incorrectly identified during flooding.

The reliability of this method was previously assessed through the manual inspection of a sample of buildings in Google Street View (M. Pronk, personal communication, December 2nd, 2020). However, the overall accuracy is currently unknown. Nevertheless, the use of a single height reference and average storey height for all buildings suggests that this approach would be unable to provide accurate results in all cases. Part of this thesis will therefore focus on evaluating the performance of this method (see Section 5.6).

2.2.2 Area-based approach

The area-based approach provides an estimate of the number of floors by dividing the internal floor area of a building by its footprint area (Figure 2.4b). In the Netherlands, the Net Internal Area (NIA) of each building is documented within the BAG and can be used for this purpose. The NIA does not represent the total internal floor area of a building, but rather the usable floor area within a building. This means that, among other things, it excludes areas related to walls, stairs, elevator shafts and places where ceiling

height is lower than 1.5 metres (Figure 2.5). However, underground floor area is included in the NIA, meaning that underground storeys could be inferred [Biljecki et al., 2017].

This approach is used by the `www.overstroomik.nl` application when building height information is unavailable (M. Pronk, personal communication, July 28th, 2020). For example, for buildings constructed more recently than the latest AHN point cloud. The height-based method is preferred because NIA generally provides an underestimate of the number of floors, as it only covers the usable area of a building. To reduce underestimation, the results are rounded up if the fractional part represents 10% or more of a floor. This means that a value of 2.1 is rounded to 3 floors (M. Pronk, personal communication, July 28th, 2020). However, the area-based approach remains less reliable because NIA is often incorrectly registered in the BAG. There are also discrepancies in its calculation before and after the introduction of the Dutch standard for areas and volumes of buildings (NEN 2580:2007) [Boeters, 2013]. In addition, some footprints are much larger than the rest of the building above or even represent multiple buildings with differing floor counts.

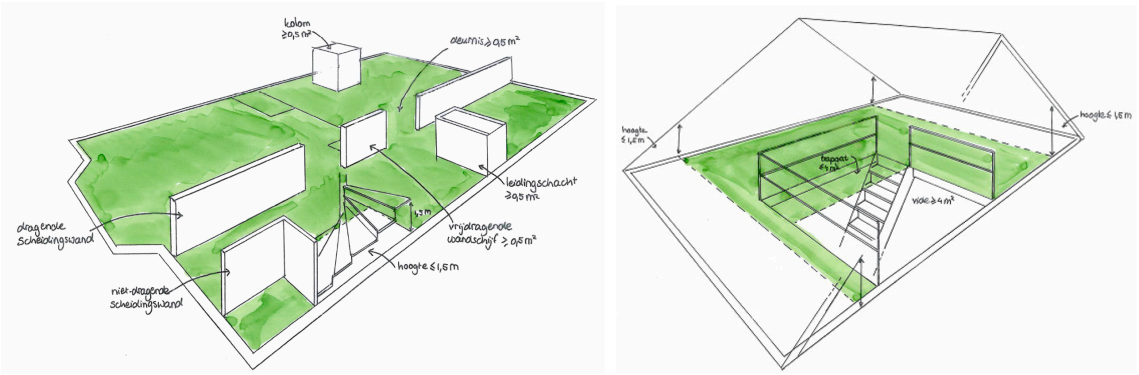


Figure 2.5: Illustration of the surface areas included (green) and excluded (white) from the NIA. Adapted from Ellenkamp and Rietdijk [2010].

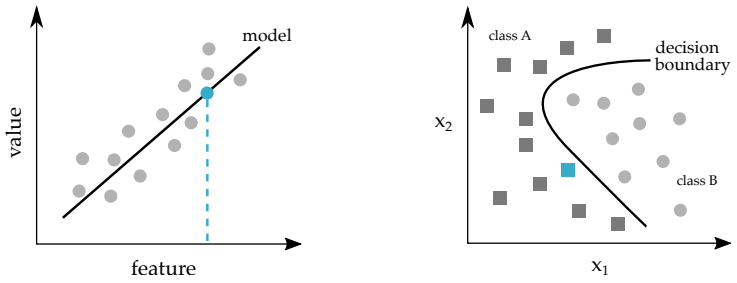
2.3 Machine learning

2.3.1 General background

Machine learning is the field of study that enables computers to learn from data without being explicitly programmed (Arthur Samuel, 1959, as cited in Géron [2019]). The data that a computer learns from is called *training data*. This data is used to generate mathematical models, based on which predictions or decisions are made. Since machine learning systems do not rely on hard-coded rules, they can be used to gain insights into large, complex datasets or to solve problems for which conventional approaches do not perform well. It is a technique used for a wide variety of tasks ranging from email spam filters to facial recognition systems [Géron, 2019].

Machine learning techniques can be divided into different categories. One major distinction is made between *supervised* and *unsupervised* methods. Supervised algorithms require the training data to include the desired solutions, called labels. The algorithm then creates a mapping between the desired labels and the rest of the input data, known as features. Unsupervised algorithms require only features and determine the mapping to labels independently. For instance, by clustering the data into similar groups. This thesis focuses on inferring the number of floors using supervised techniques. The labels are the building floor count and the features are building properties, such as roof shape and height.

Supervised algorithms can be further sub-divided into *regression* and *classification* tasks (Figure 2.6). Regression algorithms are used to predict continuous values (e.g. building height), whereas classification algorithms are used to predict discrete classes (e.g. roof shape categories) [Müller and Guido, 2016].



(a) Regression (b) Classification

Figure 2.6: Two main types of supervised algorithms (predictions for new instances shown in blue)

Initially it was unclear whether inferring the number of floors was a regression or classification problem. This is because floor count is generally described by integer values, which could be predicted as discrete classes by a classification algorithm or obtained after rounding the predictions of a regression algorithm. However, classification would require the training data to include examples of all possible floor counts that exist in reality. In practice, it would be difficult to find this data, meaning that the model would be unable to predict the number of floors of the missing classes. In contrast, the predictions of a regression model are not limited to the training data examples, allowing predictions to be made for unseen cases. The model takes into account the ordinal nature of the number of floors and would remain applicable to new buildings with higher floor counts. Therefore, it is more appropriate to consider this a regression problem. A further advantage of regression is that floor count is predicted as a decimal value, allowing buildings with half floors to be taken into account. This is required for certain applications, such as energy demand estimation.

The following sub-sections outline the three regression algorithms used in this thesis.

2.3.2 Random Forest Regression

The first machine learning algorithm used in this thesis is Random Forest (RF) Regression. RF is an *ensemble* method first introduced by Breiman [2001]. Ensemble methods combine the predictions of several base models in order to provide better predictions than an individual model alone. In the case of RF, the algorithm works by combining multiple Decision Trees [Géron, 2019]. A Decision Tree is a machine learning algorithm with a tree-like structure. It essentially forms a hierarchy of if/else questions, which the algorithm constructs by finding the best feature and threshold value to split each node [Müller and Guido, 2016]. The main drawback of Decision Trees is their tendency to construct decisions that are highly sensitive to the input training data, leading to models with high variance and low bias. This makes it difficult for Decision Trees to generalise well to new (unseen) data, a problem called overfitting. By combining many slightly different trees into a RF, overfitting can be averaged out, allowing more reliable results to be obtained. A schematic representation of RF is shown in Figure 2.7.

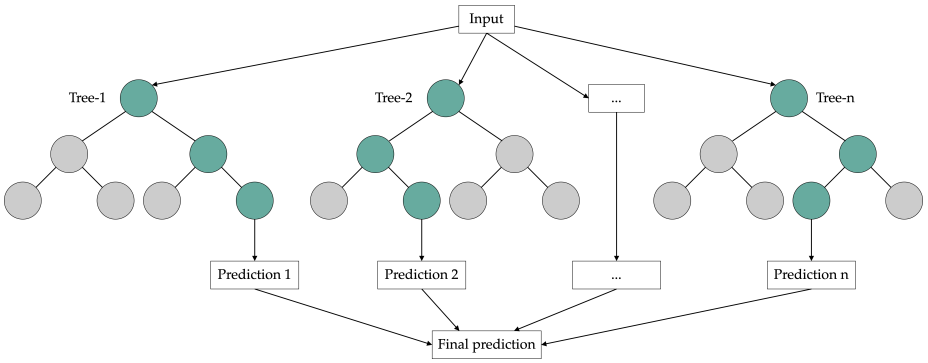


Figure 2.7: Schematic representation of Random Forest

In order to create an ensemble consisting of many slightly different trees, each tree is trained on a different random subset of the training data. The algorithm does this using a process called bootstrap aggregating or *bagging*, which involves randomly sampling the training data with replacement [Breiman, 1996]. Furthermore, rather than determining the best feature to split each node, the algorithm determines the best feature from a random subset of features. This prevents the trees from becoming correlated to each other. As a result, each tree is different and overfits the training data in a different way [Müller and Guido, 2016].

A final prediction is made by aggregating the predictions of the individual trees and averaging the result over the number of trees. This leads to a model with similar bias to the individual trees but lower variance, meaning overfitting is reduced [Géron, 2019]. The prediction made for an unseen sample x' is shown in Equation 2.1, in which the predictions of the individual trees is represented by $f_t(x')$ and the total number of trees by T.

$$\hat{f} = \frac{1}{T} \sum_{t=1}^{T} f_t(x') \tag{2.1}$$

A useful property of RF is that *feature importance* is computed as part of the training process. Each feature receives a score between 0 and 1, with a higher score representing a higher importance in the decision-making process. This is calculated by determining the mean decrease in *impurity* of nodes that use each feature. In regression problems, variance is often used as the impurity measure. The more a feature reduces impurity, the more important it is. The impurity score is also weighted by the number of training samples associated with each node, meaning that features used to generate splits higher up in the tree are generally more important [Géron, 2019]. The sum of the feature importances equals 1, meaning the score reflects the relative importance of each feature. This allows a better understanding of the factors influencing the problem to be gained, increasing the interpretability of the model. Furthermore, feature importance can be used to select the best subset of features to train a model on, helping to reduce model complexity [Louppe, 2015; Grömping, 2009].

A further useful characteristic of RF is that it is a non-parametric model. This means that it does not have a predetermined number of parameters that limit its degree of freedom to fit the training data. As a result, the model does not make any assumptions about the relationship between the features and target values, making it effective for both linear and non-linear distributions. Without any restrictions, this can lead to overfitting [Géron, 2019]. However, for inferring the number of floors, this characteristic is useful because the type of relationship between floor count and other building properties is unknown.

2.3.3 Gradient Boosting Regression

The second algorithm used in this thesis is Gradient Boosting (GB) regression, which is another ensemble method first introduced by Breiman [1997] and further developed by Friedman [2001]. More specifically, this thesis uses a GB regression based on Decision Trees, an algorithm known as Gradient Boosted Regression Trees (GBRT). A *boosting* algorithm is any ensemble method that combines many simple models, known as *weak learners*, into a model that achieves high accuracy, known as a *strong learner*. Boosting algorithms generally create a strong learner by sequentially adding predictors to an ensemble, with each new predictor attempting to correct its predecessor. In the case of GB regression, each new predictor is fitted to the residual errors of the previous [Géron, 2019]. Unlike RF, the algorithm does not rely on introducing randomness to the model to ensure each tree is different. Instead, the maximum depth of each tree is restricted to create an ensemble of shallow trees that each provide a good fit to a particular part of the data [Müller and Guido, 2016].

An example of GBRT is shown in Figure 2.8. The models generated by three sequential predictors are shown on the left and the resulting ensemble's prediction on the right. Since the ensemble consists of only one tree in the beginning, the first ensemble prediction is the same as the first tree's prediction. In the next two training steps, a new predictor is trained on the residual errors of the previous predictor. This results in an ensemble prediction that gradually improves with each additional training step. The

ensemble's prediction is equal to the sum of the individual predictions made by the model's constituent trees. This is shown by Equation 2.2, in which the prediction made by each tree is represented by $f_t(x')$ and the total number of trees by T.

$$\hat{f} = \sum_{t=1}^{T} f_t(x') \tag{2.2}$$

Similar to RF, GBRT have the ability to determine feature importances. Furthermore, since the algorithm is also based on Decision Trees, it is non-parametric and does not make assumptions about the type of relationships present in the training data. Therefore, the characteristics of RF that may be useful for inferring the number of floors are also applicable to GBRT.

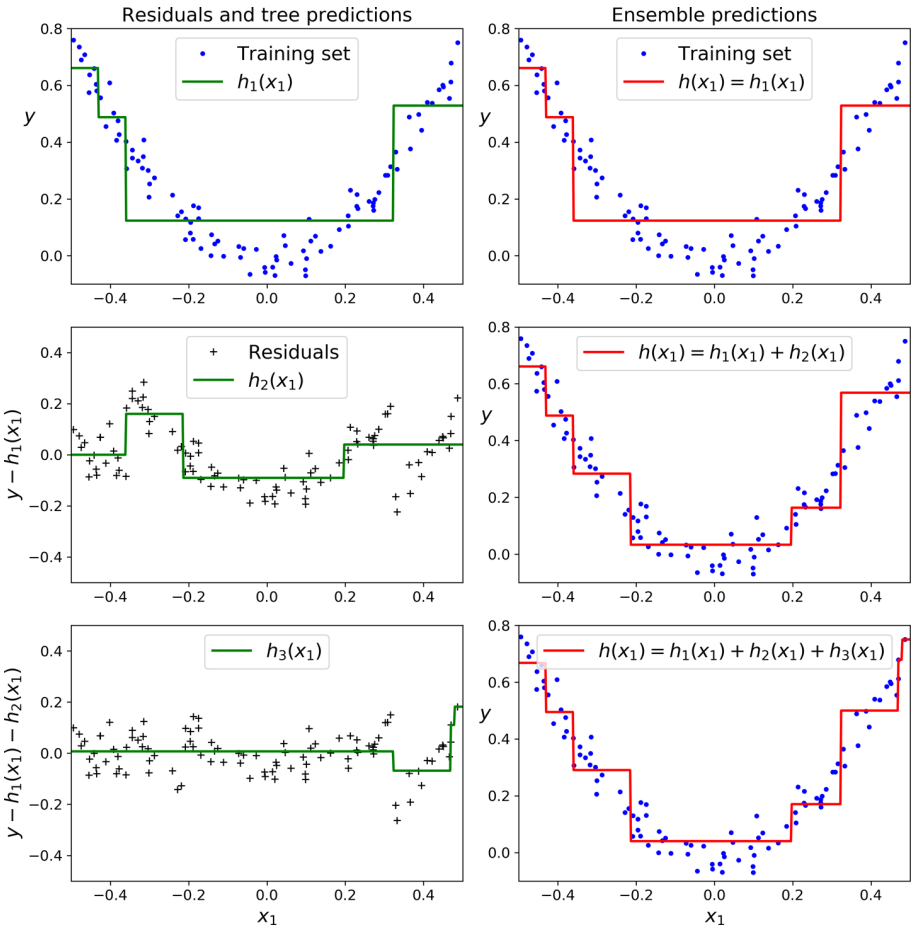


Figure 2.8: Example of Gradient Boosting regression showing the model generated by each individual predictor (left) and the resulting ensemble predictions (right). From Géron [2019].

2.3.4 Support Vector Regression

The final algorithm used in this thesis is Support Vector Regression (SVR). This is a generalisation of Support Vector Machines (SVMs), which were originally developed for classification problems. The aim of SVR is to determine the function $f(x)$ that does not deviate more than ϵ from the labels y_i of the training data, while also being as flat as possible [Smola and Schölkopf, 2004]. This is shown in Figure 2.9 for a linear function.

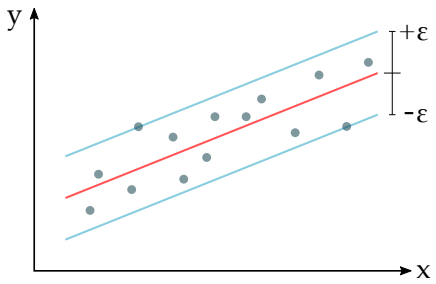


Figure 2.9: Linear Support Vector Regression. Adapted from Smola and Schölkopf [2004].

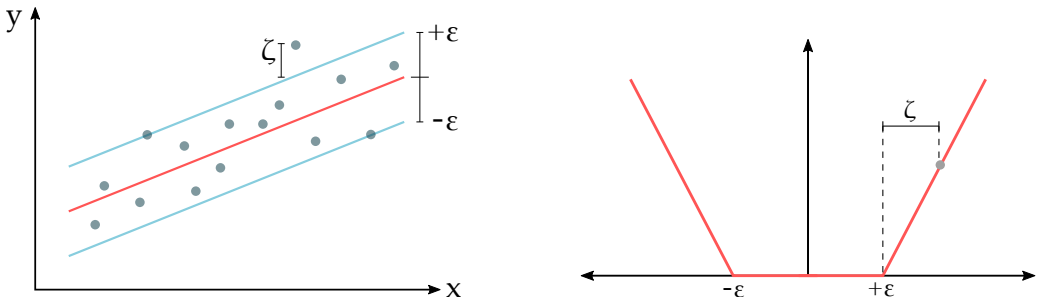
In mathematical terms, the linear case of $f(x)$ takes the form shown by Equation 2.3, in which w is a vector of weights assigned to the training instances x and b is the intercept.

$$f(x) = w^T \cdot x + b \tag{2.3}$$

In order to ensure the function is as flat as possible, w should be small, which can be achieved by minimising its norm. This allows the aim of SVR to be formulated mathematically as in Equation 2.4.

$$\begin{aligned} &\text{minimise} && \frac{1}{2}||w||^2 \\ &\text{subject to} && \begin{cases} y_i - w^T \cdot x - b \leq \epsilon \\ w^T \cdot x + b - y_i \leq \epsilon \end{cases} \end{aligned} \tag{2.4}$$

In some cases, it may not be feasible to satisfy these conditions. In other words, there may not be a linear function that can fit the training data within a tolerance of ϵ. In order to allow for such cases, the model constraints can be softened by introducing so-called slack variables ζ_i, ζ_i^*, as shown in Figure 2.10a.



(a) Soft margin setting

(b) ϵ-insensitive loss function

Figure 2.10: Linear Support Vector Regression with slack variables.
Adapted from Smola and Schölkopf [2004].

Including the slack variables in the original formulation leads to Equation 2.5. The constant C (> 0) determines the balance between the flatness of the function f and the degree to which deviations greater than ϵ are allowed [Smola and Schölkopf, 2004].

$$\begin{aligned} &\text{minimise} && \frac{1}{2}||w||^2 + C\sum_{i=1}^{\iota}(\zeta_i + \zeta_i^*) \\ &\text{subject to} && \begin{cases} y_i - w^T \cdot x - b & \leq \epsilon + \zeta_i \\ w^T \cdot x + b - y_i & \leq \epsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* & \geq 0 \end{cases} \end{aligned} \tag{2.5}$$

This corresponds to the ϵ-insensitive loss function shown visualised in Figure 2.10b. Training instances within the ϵ-margins are ignored by this function. The loss measure is based on the distance between the training instance and the closest ϵ-margin and increases linearly with distance from this margin. The ϵ-insensitive loss function is defined by Equation 2.6.

$$|\zeta|_\epsilon := \begin{cases} 0 & \text{if } |\zeta| \leq \epsilon \\ |\zeta| - \epsilon & \text{otherwise} \end{cases} \tag{2.6}$$

The SVR algorithm can also be applied to non-linear regression problems through the use of a non-linear kernel function. The kernel function allows the input data to be mapped to a higher dimensional feature space, meaning that that the standard SVR algorithm presented above can be applied [Smola and Schölkopf, 2004]. However, for the purposes of this thesis, the focus is on linear SVR because the fit time complexity of using non-linear kernels is more than quadratic with the number of samples. This makes it unsuitable for training datasets larger than ten thousand samples [Scikit-learn, 2021d], which is the case in this thesis. In addition, the use of a linear SVR will make it possible to assess whether a purely linear algorithm is sufficient to provide accurate predictions.

2.4 Machine learning and 3D building models

A number of previous studies have successfully used machine learning to generate and enrich 3D building models from building footprints and attributes. This provides a useful reference for predicting the number of floors from similar data.

Biljecki et al. [2017] used machine learning to infer building heights without elevation data for 200,000 buildings in Rotterdam, the Netherlands. Three categories of features were used in this analysis. The first was based on cadastral data and consists of features such as construction year, building function and number of floors. The second category was based on census data, consisting of demographic and socio-economic indicators such as population density and average income. This data was available at a neighbourhood level rather than per building. Finally, the third category was based on the geometry of the building footprints and included features such as area, perimeter and shape complexity. Different combinations of these features were used as input to a RF regression algorithm to represent different levels of data availability.

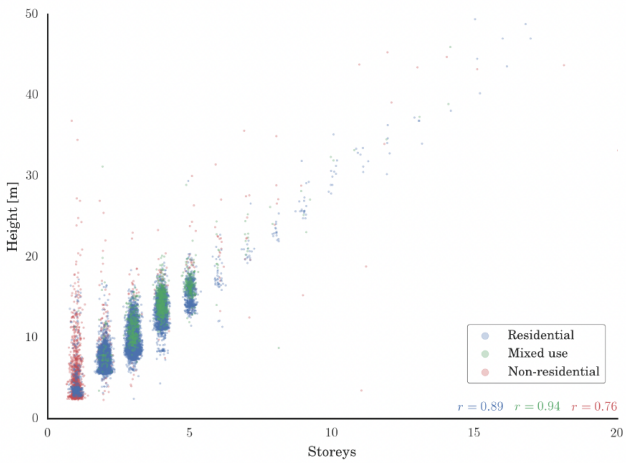


Figure 2.11: Relationship between height and number of floors, showing that the relationship is strongest for residential buildings. Reprinted from Biljecki et al. [2017]

The number of floors, building age and net internal area were found to be the features with the highest importance. A strong linear correlation was observed between height and the number of floors, particularly for (mixed-)residential buildings (Figure 2.11). As a result, it seems likely that building height,

and potentially also building age and net internal area, would be important for inferring the number of floors from similar data. The best model provided a higher level of accuracy than the commonly used geometric approach, which is based on multiplying the number of floors by an assumed storey height (the inverse of the approach presented in Section 2.2.1). This suggests that machine learning could also be a more powerful way to infer the number of floors than the current geometric approach.

Lánský [2020] further extended this research by inferring the height of all buildings in the USA, using similar features derived from 2D building footprints and attributes. The level of accuracy achieved for rural and suburban areas was relatively good. However, the error in central business districts was high, potentially due to the training data not being representative enough for this area morphology. Additional non-geometric features helped to reduce the prediction error for suburban areas, but obtaining accurate predictions for central business districts remained difficult.

Similar research has also been conducted by Anh et al. [2018] and Milojevic-Dupont et al. [2020]. Anh et al. [2018] used the same geometric predictors as Biljecki et al. [2017] to infer the height of buildings in Hanoi, Vietnam. However, the performance of the model was lower, which the authors attribute to the limited amount of training data used by the algorithm. Milojevic-Dupont et al. [2020] used 152 different geometric features based on data on urban form, such as building footprints and street networks. The analysis found that the morphology of the urban fabric surrounding a building was highly predictive of building height. The average error of the best model was substantially lower than the average storey height of 2.5m and the model predictions generalised well across countries. However, larger prediction errors were obtained for tall buildings [Milojevic-Dupont et al., 2020].

In terms of model enrichment, machine learning has been used to infer a variety of building attributes including building type and age [Biljecki and Sindram, 2017; Henn et al., 2012]. Biljecki and Dehbi [2019] explored the use of machine learning to infer roof shape from LOD0 and LOD1 models without the acquisition of LiDAR data. The model achieved an accuracy of 85% for the prediction of six roof classes and 92% for distinguishing flat roofs from slanted roofs, showing that machine learning is a powerful way to enrich 3D building models This research is closely related to the previously described work of Biljecki et al. [2017], as it forms part of a possible pipeline for constructing LOD2 models from footprints without elevation data (Figure 2.12). Using machine learning to further enrich the model with information on the number of floors could potentially also fit into this pipeline.

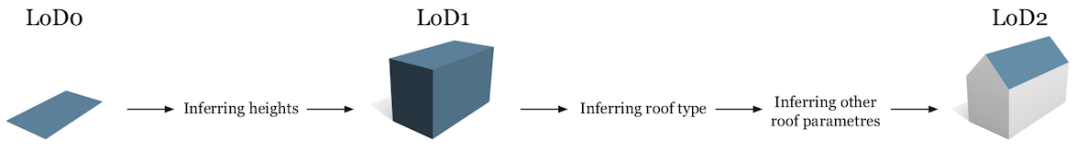


Figure 2.12: Pipeline for generating LOD2 models from building footprints without elevation data. Adapted from Biljecki and Dehbi [2019].

The use of machine learning to infer the number of floors has already been investigated in a recent study by Krayem et al. [2021]. A multi-layer feed forward Neural Network (NN) was used to infer the number of floors of 6877 buildings in Beirut, Lebanon. The training dataset consisted of 1536 buildings and four input features: building area, perimeter, height and electricity consumption. A correlation analysis was used to show that the number of floors was linearly dependent on these features, particularly building height. The final model achieved a high level of accuracy and the average error was approximately half a storey. These results provide an interesting comparison for the analysis performed in this thesis, which is conducted on a larger scale, using more input features and less complex algorithms.

3 Methodology

This chapter outlines the methodology used to address the research objectives. As shown in Figure 3.1, the methodology consists of four main stages: (1) data collection and integration, (2) data preparation, (3) modelling and prediction and (4) model assessment. The following sections describe the steps involved in each stage in further detail.

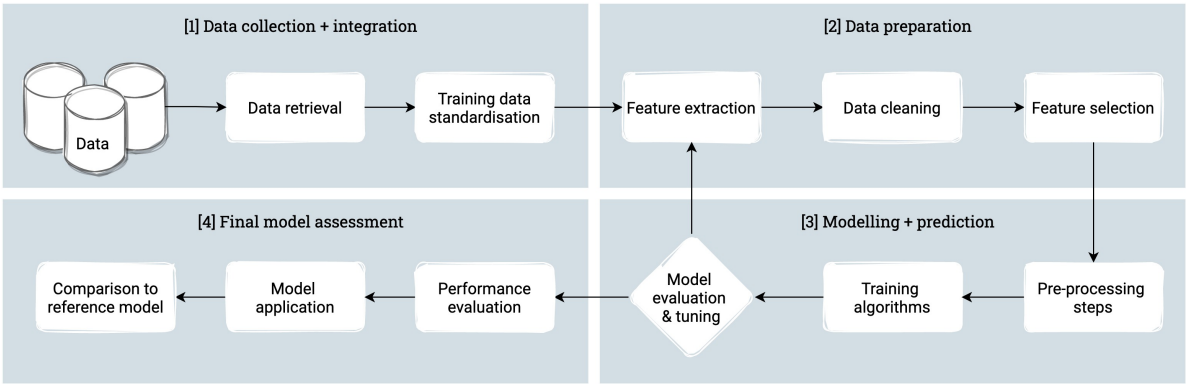


Figure 3.1: Flowchart of methodology

3.1 Data collection and integration

The first stage of the methodology involved collecting and integrating the data required for the analysis. Firstly, data on the number of floors was obtained from different municipalities. This data was used to create a basic training dataset consisting of unique building identifiers and floor count. Since each municipality has a different approach towards storing this data, a standardisation step was required before the data from each source could be integrated. Part of this standardisation step also involved automatically cleaning the data to remove any obvious errors, such as cases where the number of floors was zero. In addition, data on each building's current use was obtained from the BAG and used to filter for (mixed-)residential buildings. All non-residential buildings were discarded, helping to reduce data storage requirements and processing times. Further details on the training data sources and automatic cleaning steps are provided in Section 4.1 and Section 4.3 respectively.

3.2 Data preparation

3.2.1 Feature extraction

The second stage of the methodology involved preparing the data for the training process. The first step was to extract features that describe the properties of each building. In total, 25 features were extracted. These features were obtained from three main datasets and can be subdivided into cadastral, geometric and census-based features. Further details on the datasets used are provided in Appendix A.

Cadastral features

The cadastral features are summarised in Table 3.1, alongside their relevance to the number of floors. These features were obtained from the BAG. As explained in Chapter 1, this dataset consists of polygons representing the footprints of all buildings in the Netherlands. It also consists of points representing all building units. These units can be linked to each building by the footprint ID, which acts as a foreign key. Each building footprint and unit are associated with a number of attributes. Construction year could be obtained directly since it is a building footprint attribute, while the other features were obtained by summing the values of all units associated with a building.

Table 3.1: Cadastral features

	Feature	Details and relevance
1	Construction year	Construction period is often related to storey height. For instance, after 2003, the Dutch building code increased the required storey height of new buildings from 2.4 to 2.6 meters [Ministry of the Interior and Kingdom Relations, 2012]. This means that construction year could be used to distinguish buildings with the same number of floors but different heights.
2	Building function	A distinction is made between residential and mixed-residential, as mixed-residential buildings have been found to exhibit different properties than purely residential buildings [Biljecki et al., 2017].
3	Net internal area	Previous research has found that taller buildings (with more storeys) generally have a higher net internal area [Biljecki et al., 2017].
4	Number of units	Similar to the net internal area, buildings with more storeys generally contain more building units (e.g. apartment blocks).

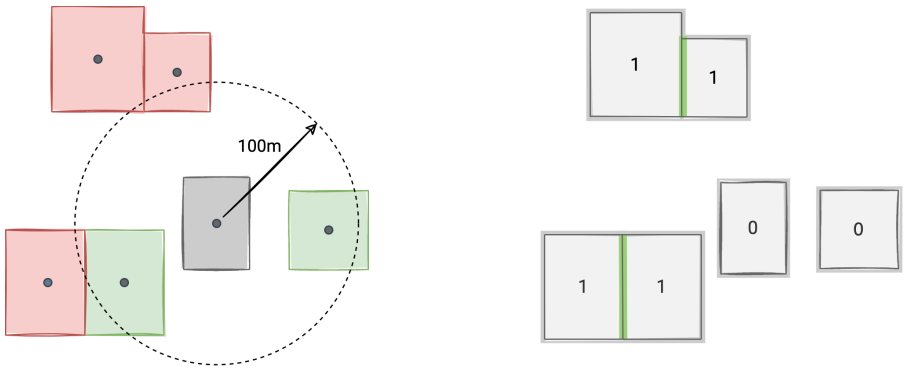
Geometric features

The geometric features can be further subdivided into 2D and 3D features. The 2D features were extracted from the BAG building footprints and are summarised in Table 3.2. These were based on the features used by Lánský [2020] to infer building height with machine learning. The first three (area, perimeter and number of vertices) were extracted to describe the characteristics of the footprint shape. Footprint area could also provide an indication of the number of floors when combined with net internal area, as discussed in Section 2.2.2. In order to ensure that the number of vertices is representative of the building shape, the Douglas-Peucker algorithm was used to simplify the footprint boundary and remove any collinear points.

Table 3.2: 2D geometric features

	Feature	Details and relevance
5	Area	Dividing the net internal area by the footprint area can provide an indication of the number of floors (see Section 2.2.2).
6	Perimeter	In combination with area, perimeter can provide information about the footprint shape, such as its compactness and complexity [Lánský, 2020].
7	No. vertices	A higher number of footprint vertices could indicate a more complex shape [Lánský, 2020]. Computed after simplification by Douglas-Peucker.
8	No. neighbours	The number of neighbouring building centroids within a 100m radius of the footprint centroid. Buildings with many storeys are generally surrounded by more open space [Biljecki et al., 2017]. Buildings in rural areas also generally have fewer neighbours [Lánský, 2020].
9	No. adjacent buildings	The number of buildings within a 0.1m buffer of each footprint. Lower storey buildings in urban areas generally have more immediate neighbours.

Features 8 and 9 were added to provide information on the building surroundings. The number of neighbours was defined as the number of other footprint centroids within a 100 meter radius of each footprint centroid (Figure 3.2a). A radius of 100 meters was selected based on experiments (see Section B.1). The centroid method was used because it was less computationally expensive than computing buffer intersections for all footprints. This allows for better scalability if the model was applied to the whole country [Lánský, 2020]. However, the drawback is that the results for large building footprints may be lower than reality, as the radius contains the distance from the centroid to the footprint boundary. In order to provide information about the more immediate vicinity of a building, the number of adjacent buildings was computed. A buffer of 0.1 meters was created around each footprint and the number of adjacent buildings was defined as the number of intersections with this buffer. As shown in Figure 3.2b, this provides a measure of the number of walls shared with other buildings.



(a) Number of neighbours (b) Number of adjacent buildings

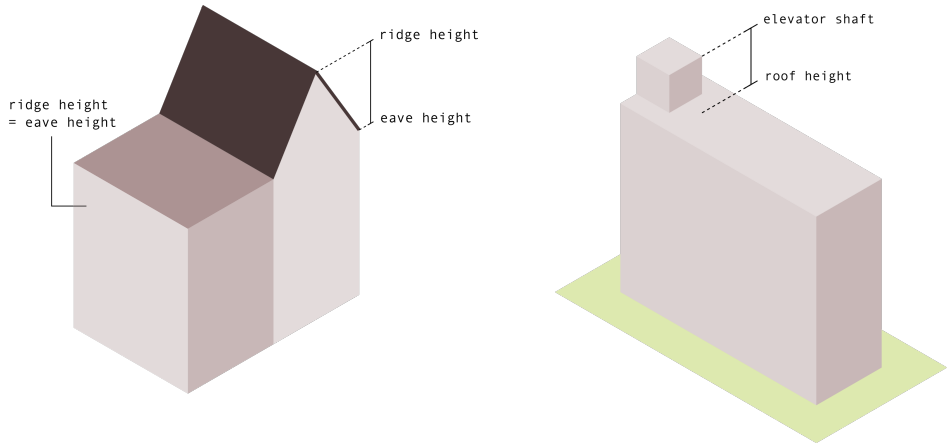
Figure 3.2: Computation of neighbouring and adjacent buildings. Adapted from Lánský [2020].

The 3D geometric features are summarised in Table 3.3. These features were extracted from the LOD1.2 and 2.2 models available for all buildings in the 3D BAG. As explained in Chapter 1, this is a 3D version of the BAG, created by combining each building footprint with height information from the AHN point cloud to automatically generate 3D building models at different levels of detail. This dataset also consists of a number of attributes, such as roof shape, ground height and roof height at four percentiles of the z-coordinates of the building's point cloud. These different percentiles represent different height references for the roof, as discussed in Section 2.1.2. Building height was computed by subtracting ground height from the different roof height percentiles.

Table 3.3: 3D geometric features

	Feature	Details and relevance
10	Building height	Computed for the minimum, maximum, 50th and 70th roof height percentiles (available as attributes of the 3D BAG). Building height is strongly related to number of floors, especially for residential buildings [Biljecki et al., 2017].
11	Roof shape	An attribute provided for each building in the 3D BAG. In combination with building height, roof shape could provide information about the likelihood that storeys are present beneath slanted roofs.
12	Ridge vs. eave height	The difference between the height of the ridge and eaves of the roof. Similar to roof shape, this could provide some indication of whether storeys might be present beneath slanted roofs (Figure 3.3a).
13	Roof surface area	Computed for both LOD1.2 and LOD2.2 to describe building geometry.
14	Wall surface area	Computed for both LOD1.2 and LOD2.2 to describe building geometry.
15	Building volume	Computed for both LOD1.2 and LOD2.2. A larger volume is somewhat linked to a larger number of floors.

The other four features were calculated from the geometry of the 3D models. The difference between ridge and eave height was based on the LOD2.2 model. It was computed for buildings with slanted roofs to provide an indication of any storeys below the roof (Figure 3.3a). The concept was also extended to buildings with multiple horizontal roof surfaces, to potentially allow buildings with elevator shafts and other roof installations to be distinguished (Figure 3.3b). These structures increase the measured height of a building, but should not contribute to the floor count. Roof surface area, wall surface area and building volume were computed from both the LOD1.2 and 2.2 models. These features were extracted for multiple levels of detail in order to determine whether a higher LOD provides better results than a simpler model. Further details on the extraction of these features can be found in Section 4.4.



(a) Slanted and single horizontal roofs (b) Multiple horizontal roof

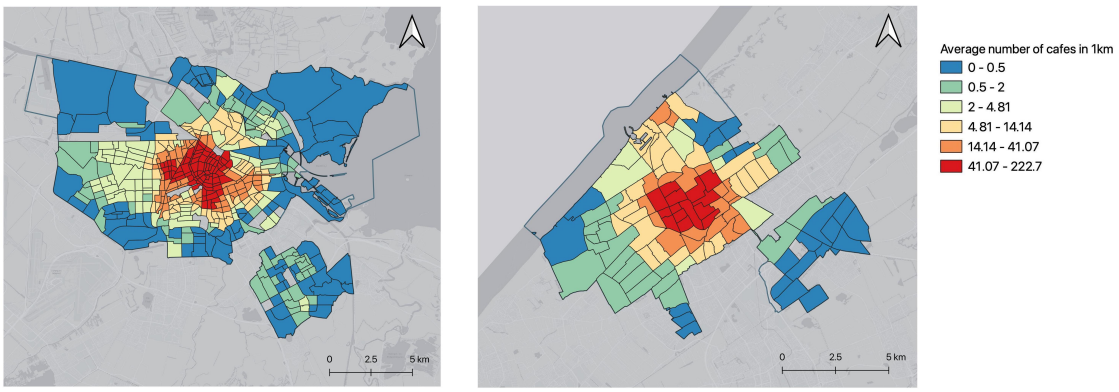
Figure 3.3: Ridge vs. eave height based on roof type

Census features

The census features are outlined in Table 3.4. These features were obtained from a neighbourhood census dataset maintained by the *Centraal Bureau voor de Statistiek (CBS)*, a government agency responsible for collecting statistical information about the Netherlands. Since this dataset is only available at a neighbourhood level, buildings from the same statistical neighbourhood received the same value for each feature. Each statistical neighbourhood covers many different types of buildings, with a large range in storeys, meaning that the direct relationship to the number of floors is diminished. Nonetheless, these features could help to improve the accuracy of the results when used in combination with other features by providing information about the building's surrounding environment. For example, higher storey buildings accommodate more people, which results in a higher demand for amenities. As a result, data on amenities could help to improve the predictions [Biljecki et al., 2017].

Table 3.4: Census features

	Feature	Details and relevance
16	Population per km^2	Areas with a higher population density generally have more high storey buildings to accommodate all residents.
17	Percent multi-household	Multi-household buildings, such as apartment blocks, generally have more storeys than single family homes.
18	Average no. of cafes in 1km	The average number of cafes shows a strong link to area morphology (Figure 3.4) and could be used to distinguish central business districts from rural and suburban areas. Other amenities were also considered but the average number of cafes showed the clearest relationship to area morphology (see Section B.2).



(a) Amsterdam (b) Den Haag

Figure 3.4: Average number of cafes in 1km per statistical neighbourhood

Additional features

An additional feature was obtained from a dataset on building type maintained by ESRI [ESRI, 2021]. This dataset uses the geometry of the BAG building footprints in combination with the number of addresses associated with each footprint to determine building type. This feature was added to provide more information on building form and consists of four values: detached, semi-detached, terraced and apartment blocks.

Table 3.5: Additional features

	Feature	Details and relevance
19	Building type	Apartment blocks generally have a higher number of floors than (semi-)detached and terraced buildings.

3.2.2 Data cleaning

After extracting the features, the second data preparation step involved data cleaning. This step was required in order to ensure that the machine learning algorithms would be trained on reliable data. If a substantial amount of erroneous data was used as input, the accuracy of the model predictions would be reduced. The cleaning steps mainly focused on validating the training data labels and removing any cases where the number of floors had been incorrectly labelled. In some cases, the training labels were manually corrected. However, since the size of the training dataset was quite large, the cleaning process focused on automatic and semi-automatic steps to filter out gross errors. Further details on the implementation of the data cleaning steps is provided in Section 4.3.

3.2.3 Feature selection

The final data preparation step focused on determining which features contributed most to the prediction problem, with the aim to eliminate any redundant or irrelevant features. This process is known as *feature selection* or *feature elimination*. The purpose of feature selection is to select a subset of features that can provide a concise description of the training dataset, while still generating accurate predictions [Chandrashekar and Sahin, 2014].

Feature selection is beneficial for a number of reasons. Firstly, it reduces the number of input variables that the model has to fit, which lowers the computational cost of training. Secondly, only the most relevant features are used and less useful features are removed, reducing noise in the training data.

Furthermore, the performance of the model may be improved since the algorithm is not focused on fitting the model to irrelevant features, which could lead to an overfit of the training data. Finally, the complexity of the model is reduced, which allows a better understanding of the data to be obtained [Guyon and Elisseeff, 2003; Chandrashekar and Sahin, 2014].

A variety of approaches can be used to perform feature selection, each leading to potentially different results. In this thesis, three approaches were used in order to consider different aspects of the feature selection problem. The details of these approaches and their results are discussed in Section 4.5.

3.3 Modelling and prediction

3.3.1 Pre-processing

The next stage of the methodology involved training the machine learning algorithms to generate different predictive models. Before the algorithms could be trained, a number of pre-processing steps were required. Firstly, a test set was created by setting aside 20% of the training data, as illustrated in Figure 3.6a. This data was kept aside during the training process so that the model could be evaluated on unseen data. Evaluating the model on unseen data allows a more reliable measure of model performance to be obtained and also makes it possible to determine whether the model over- or under-fits the training data.

The other pre-processing steps focused on converting the training features to an appropriate format for the machine learning algorithms. Numerical and categorical features were processed separately. The first step was to handle features with missing values, since the algorithms cannot be trained on incomplete data. Rather than discarding any incomplete training instances, missing values were *imputed*, meaning they were inferred from the known part of the data. For categorical features, a simple imputation strategy was used in which missing values were replaced with the most frequent value per feature. For numerical features, a more sophisticated approach was used through which all other available features were used to estimate the imputed value. Further details are provided in Section 4.3.1.

The next step was to convert the categorical features to a format that the algorithms could interpret. In order to do this, *one-hot encoding* was applied. As shown in Figure 3.5, one-hot encoding involves creating an additional column for each possible feature value. Each additional column represents a new feature, which has a value of 1 when it applies to the training instance and 0 otherwise. If many categorical features are present or if each categorical feature has many possible values, this leads to many additional features and a corresponding increase in model complexity. However, this approach is preferred over simply converting categorical features features to integer values. This is because machine learning algorithms interpret consecutive numbers as being ordered, which was not appropriate for the features present in this training dataset.

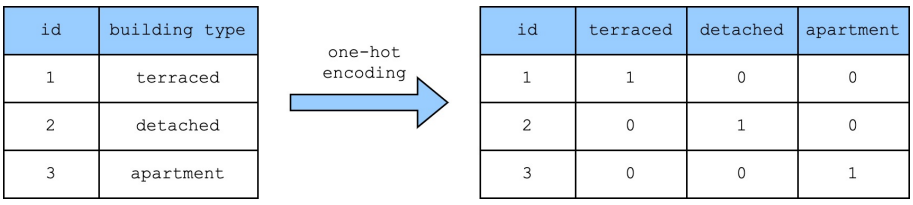


Figure 3.5: Example of one-hot encoding of categorical features

The final pre-processing step was to apply *feature scaling* to the numerical features. The features were normalised by subtracting the mean and scaling to unit variance. This process is defined by Equation 3.1 where x' is the scaled feature, $\hat{x}$ the mean and σ the standard deviation. Feature scaling was performed because SVR performs poorly if the input features do not have similar scales [Géron, 2019].

Tree-based algorithms (RF and GB) do not require feature scaling but, since the results are not affected by the magnitude of the input variables, the same pre-processing steps were applied.

$$x' = \frac{x - \hat{x}}{\sigma} \tag{3.1}$$

3.3.2 Training, evaluation and tuning

After performing the pre-processing steps, three different algorithms were trained: Random Forest (RF), Gradient Boosting (GB) and linear Support Vector Regression (SVR). The training process was performed using an iterative approach and some data preparation steps, such as data cleaning, were repeated throughout the process.

Firstly, each algorithm was trained on all available features and the resulting models were evaluated. This provided a baseline to compare the performance of future models to. During model evaluation, the predictions were rounded to the nearest integer so that they could be fairly compared to the training labels, which were provided in whole floors. Then, a combination of four error metrics were used to compare the predictions to the labels: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), maximum error and accuracy. These metrics were computed according to Equation 3.2, Equation 3.3, Equation 3.4 and Equation 3.5 respectively, in which n is the number of training instances, y_i the training labels and $\hat{y}_i$ the model predictions.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.2}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.3}$$

$$\text{Maximum error} = max(|y_i - \hat{y}_i|) \tag{3.4}$$

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}1(y_i = \hat{y}_i) \tag{3.5}$$

MAE provides a measure of the average absolute difference between the expected and predicted values. It increases linearly with this difference, meaning the same weight is given to all errors. In contrast, RMSE gives more weight to larger absolute errors, meaning variance is penalised. This makes it more suitable for analysing the performance of models that have a large amount of small errors and relatively few large errors, as the effect of these large errors would be masked in the MAE score. Penalising larger errors can also provide better insight into the effect of different model parameters. However, RMSE can be sensitive to outliers [Chai and Draxler, 2014]. Both MAE and RMSE have the same units as the training labels (i.e. number of floors), making them easy to interpret.

The maximum error is the largest absolute difference between the expected and predicted values. This metric was added to provide a measure of the worst possible model prediction. It also has the same units as the training labels. The final error metric, accuracy, is dimensionless and limited to values between 0 and 1, or 0 and 100 when converted to a percentage. It provides a measure of how often the model predicts exactly the same value as the training labels. Technically it should not be used to evaluate regression models since it is a classification-based error metric. However, since the training labels and predictions were both integers, accuracy was still meaningful in this case.

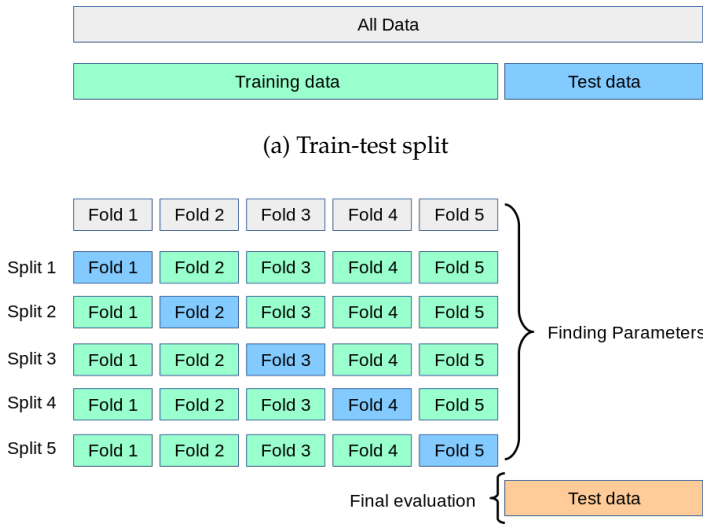
After evaluating the models based on all features, each algorithm was trained on the feature subsets chosen during the feature selection step (see Section 3.2.3). Since three different approaches were used to select the best subset of features, this led to nine different models (i.e. three per algorithm). Each of these models was evaluated using the same error metrics described above, allowing the best model per

algorithm to be selected. To further improve the performance of these three models, hyperparameter tuning was performed.

Hyperparameters are specific to each algorithm and control the learning process. For RF, hyperparameters include the number of trees in the forest and the depth of each tree. Tuning the hyperparameter values can help to improve a model's generalisation performance. This means that the algorithm will be able generate a model that provides a good fit to the training data, but will also perform well on other unseen data.

In order to tune each model, a grid of possible values for each hyperparameter was created. A randomised search was performed over 75 different combinations of these values. This means that each algorithm was trained 75 times on different combinations of the hyperparameter values provided. As a result, not all possible combinations were tested. However, this approach is preferred over an exhaustive search since this would have a much higher computational cost. Further details on the implementation of the hyperparameter tuning process are provided in Section 4.6.

To evaluate the performance of each hyperparameter combination, *k-fold cross-validation* was used, with k equal to 5. This means that the training set was further split into five smaller sets. For each fold, four of these sets were used for training and the fifth set was used to evaluate performance (Figure 3.6b). This means that each training instance was used to evaluate the model exactly once. The overall model performance was computed as the average of the five individual evaluations [Duan et al., 2003]. The model that performed best after tuning was selected as the final model.



(a) Train-test split

(b) 5-fold cross-validation

Figure 3.6: Visualisation of train-test split and cross-validation. Adapted from: Scikit-learn [2021a]

3.4 Final model assessment

The last stage of the methodology involved evaluating the final model. Firstly, a more in depth analysis of model performance was conducted based on the different error metrics described in Section 3.3.2. This included an analysis of the cumulative and gross errors. Then, the impact of rounding the predictions to the nearest integer was evaluated. After this, the importance of each feature was analysed in order to determine which features were most related to the number of floors.

Feature importance was assessed in terms of impurity importance and permutation importance. Impurity importance was only available for the tree-based algorithms (RF and GB). It is calculated by

determining the mean decrease in impurity of nodes that use each feature (see Section 2.3.2). This measure of importance suffers from a number of drawbacks. Firstly, it is derived from the training dataset, meaning it does not necessarily reflect the ability of the features to provide accurate predictions for unseen data. Secondly, it favours features with many unique values because high cardinality features have more splitting points. This causes them to be used more often in each tree, resulting in a higher importance score [Scikit-learn, 2021c].

Permutation importance does not suffer from the same drawbacks as the impurity-based importance and is applicable to any machine learning algorithm, making it a better measure of feature importance. It is defined as the decrease in model score when a random shuffle of a single feature value is performed [Breiman, 2001]. This process causes the relationship between the feature and the target variable to be broken. As a result, a decrease in model score reflects the extent to which the model depends on the feature [Scikit-learn, 2021b].

After analysing the performance of the best model, the impact of data availability was assessed. This was achieved by training the model on different subsets of features, representing different levels of data availability. For instance, a comparison was made between the performance of models trained on only LOD1.2 and LOD2.2 features, making it possible to determine to what extent a higher level of detail improved the predictions. Analysing the impact of data availability provided an alternative perspective on how different features contributed to the results. It also illustrated the potential applicability of the model to areas outside of the Netherlands, where data availability may be more limited.

In order to determine to what extent machine learning provides better results than the geometric approach, the predictions of the best model were compared to a reference model. This reference model was obtained by calculating the number of floors using the height-based approach presented in Section 2.2.1. A similar approach to the `www.overstroomik.nl` application was used. Building height was based on the 70th percentile of the roof and an assumed storey height of 2.65 meters was used. However, the results were rounded to the nearest integer rather than rounded down. A different rounding method was used because a conservative estimate was not required. Instead it was more interesting to determine how close the results were to the actual number of floors. The machine learning results were also rounded to the nearest integer, providing a fair comparison. When height was unavailable, the number of floors was calculated using the area-based approach presented in Section 2.2.2 instead.

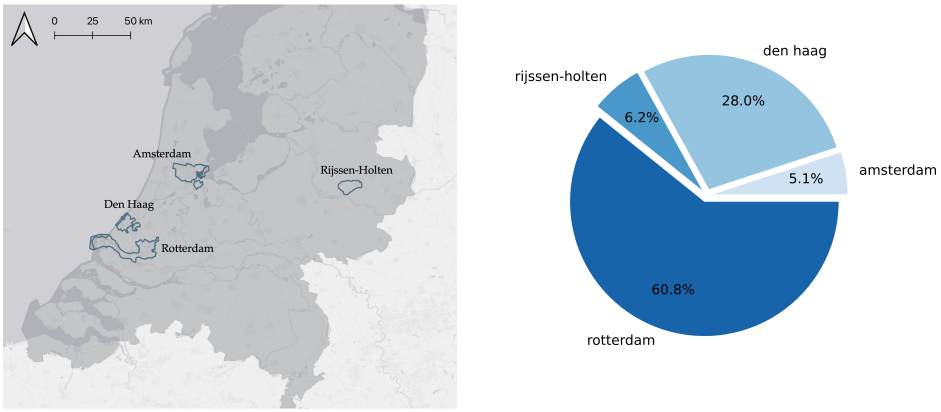
Finally, the best model was also assessed in terms of its applicability to other municipalities in the Netherlands. These municipalities were selected to represent different regions of the country, as well as different area morphologies (i.e. rural vs. urban). Since the ground truth number of floors was not available for these municipalities, the predicted number of floors of a selection of individual buildings was assessed in further detail. Google Street View was used to determine the actual number of floors of these buildings. The results were compared to the geometric approach in order to gain a further understanding of whether machine learning provided better results.

4 Implementation

This chapter presents the implementation of the methodology. Firstly, the training dataset is discussed in further detail and an overview of the programming details is provided. Then, the steps taken to clean the data are discussed, followed by the methods used to compute the 3D geometric features. After this, an overview of the three feature selection approaches is provided. Finally, the results of the hyperparameter tuning process are presented.

4.1 Training dataset

Training data on the number of floors was obtained from the BAG+ datasets of the municipalities of Amsterdam, Rotterdam, Den Haag and Rijssen-Holten (Figure 4.1a)[1]. The first three municipalities correspond to the three largest cities in the Netherlands, while Rijssen-Holten is a more rural municipality. This municipality was added to provide the algorithms with data that was also representative of less densely populated regions of the country. However, the majority of buildings originate from Rotterdam (Figure 4.1b), meaning that the dataset mainly corresponds to urbanised areas. A comparison of the different training data municipalities is provided in Table 4.1.



(a) Geographic extent (b) Data sources of residential buildings

Figure 4.1: Overview of training dataset

Table 4.1: Quantitative comparison of the training data municipalities

Municipality	No. buildings	Residential buildings (%)	Median no. floors	Area (m²)	Population density (1000 res. per km²)
Amsterdam	22,328	64	4	2.2×10^8	5.21
Rotterdam	206,809	56	3	3.2×10^8	2.96
Den Haag	53,730	99	3	9.8×10^7	6.52
Rijssen-Holten	11,879	97	3	9.4×10^7	0.41

[1]The BAG+ from Amsterdam is the only openly available dataset (accessible via their FTP-server: `ftp.data.amsterdam.nl`)

After cleaning, the training dataset consisted of 173,152 buildings ranging from 1 to 45 floors. These buildings cover a variety of architectural styles, construction periods and building types. A number of examples from each municipality are provided in Figure 4.2. These buildings are used as case studies during the analysis in order to gain a more concrete understanding of model performance (see Section 5.6). Some examples were selected to represent particularly challenging aspects of the prediction problem. For instance, buildings with varying storey heights, elevator shafts or multiple storeys beneath slanted roofs. The example shown in Figure 4.2b was selected because the building in the centre has one floor less than its neighbours due to a double height cafe on the ground floor. The geometric approach would be unable to distinguish these cases.

(a) No. floors: 4 (b) No. floors: 5–6 (c) No. floors: 15

(d) No. floors: 3 (e) No. floors: 5 (f) No. floors: 35

(g) No. floors: 2 (h) No. floors: 3 (i) No. floors: 8

(j) No. floors: 1 (k) No. floors: 2 (l) No. floors: 3

Figure 4.2: Examples of buildings in the training dataset

As shown in Figure 4.3, the training dataset is quite heavily skewed towards lower storey buildings. Data imbalance is problematic because most machine learning algorithms aim to minimise the overall error rate [Chen and Breiman, 2004]. In order to take data imbalance into account, the error metrics presented in Section 3.3.2 were computed for high and low storey buildings separately. This prevented the prediction errors for high storey buildings from being masked by the results for low storey buildings. Furthermore, a stratified approach was used to create the train-test split. This stratification was based on building height, since this is known to be highly correlated to the number of floors (Figure 2.11).

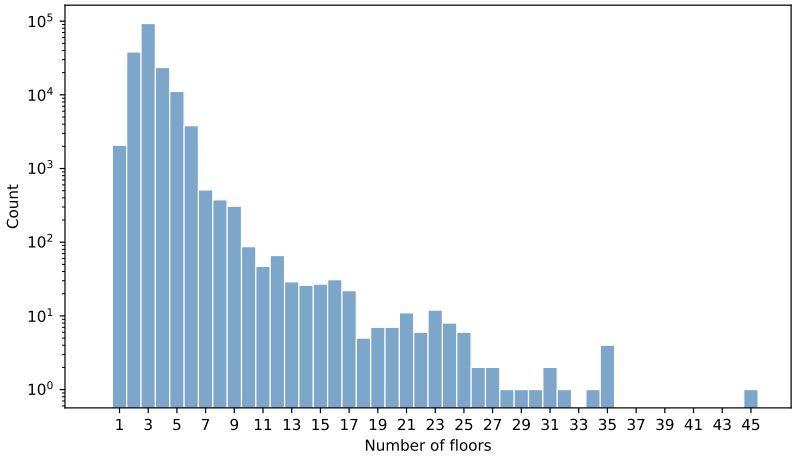


Figure 4.3: Distribution of number of floors in training dataset

4.2 Programming details

The methodology was implemented entirely in Python. All code is available at: `www.github.com/ellieroy/no-floors-inference-NL`. A PostgreSQL database extended with `PostGIS` [Strobl, 2008] was used to store the data required for the analysis. The training data from each municipality was loaded into this database using the `ogr2ogr` command line tool from the Geospatial Data Abstraction Library (GDAL) [GDAL/OGR contributors, 2021]. The other datasets were loaded into the database by restoring their database backups. This process was performed using FME, a data integration platform for spatial Extract, Transform and Load (ETL) operations.

Data preparation was partially performed in the database and partially in Python. Any steps that did not require data to be stored locally were performed in the database. This mainly involved simple operations, which could be performed using standard database functions or spatial functions available in `PostGIS v2.5.x` or higher. In order to perform these operations, queries were executed in Python using a connection to the database established with the `psycopg2` library [Gregorio and Varrazzo, 2021]. These queries were not executed directly on the training data tables, but rather on a temporary, unlogged copy of each original table. Using unlogged tables helped to improve write-performance as the data was not written to the write-ahead log [The PostgreSQL Global Development Group, 2021]. Once the queries were complete, the original tables were replaced with the contents of the unlogged tables.

The data preparation steps performed in Python covered more complex processing operations, such as the computation of the 3D geometric features and the implementation of the semi-automatic data cleaning steps. A connection to the database was established using the `psycopg2` library and used to downloaded the required data. This data was stored in a `pandas` dataframe [McKinney, 2010]. Further processing steps could then be performed on this dataframe and the results stored as new columns. After the processing steps were complete, the results were converted to a list of tuples, which were then stored in the database using the `psycopg2` method `executemany()` to perform a bulk insert.

The machine learning steps were implemented in Python using the `scikit-learn` library [Pedregosa et al., 2011]. The training data was read from the database using `psycopg2` and merged into a single `pandas` dataframe. The implementation relied on many of the built-in modules of `scikit-learn`. The pre-processing steps, such as one-hot encoding and feature scaling, were grouped together using the `DataFrameMapper` module of the `sklearn-pandas` library. This enabled all preprocessing steps to be stored in a single object and applied to the dataset using a single operation. The `Pipeline` module of `scikit-learn` was then used to combine the `DataFrameMapper` with each estimator so that preprocessing and training could be executed together. After each estimator had been trained, the model was stored to disk using the `joblib` library [Joblib Development Team, 2021], allowing predictions to be made without re-training the model.

4.3 Data cleaning steps

The data cleaning process was performed using a combination of (semi-)automatic and manual steps. An overview of the training data before and after cleaning is provided in Table 4.2. For reference, the number of (mixed-)residential buildings is also provided. In total, 59% of the raw data remained after cleaning. The largest proportion of data was removed from the datasets originating from Amsterdam and Rotterdam. However, a large percentage of the data from Rotterdam was removed because many buildings were not (mixed-)residential. If only (mixed-)residential buildings are taken into account, the data from Amsterdam was reduced most by the cleaning process. The following sections discuss the different cleaning steps in further detail.

Table 4.2: Overview of training data before and after cleaning

Municipality	Raw data	Residential data	Output data
Amsterdam	22,328	14,341 (64%)	8,757 (39%)
Rotterdam	206,809	116,638 (56%)	105,245 (51%)
Den Haag	53,730	53,559 (99%)	48,450 (90%)
Rijssen-Holten	11,879	11,516 (97%)	10,700 (90%)
Total	294,746	196,054 (67%)	173,152 (59%)

4.3.1 Automatic cleaning

The automatic cleaning steps mainly focused on removing obvious errors and outliers from the data. Cases where the number of floors was missing (i.e. zero or `NULL`), larger than 48 or less than zero were removed, as well as any duplicate buildings. The upper limit for the number of floors was set to correspond to the building with the highest number of floors in the Netherlands. Buildings with negative floor counts were removed because this thesis focuses solely on estimating the number of floors above ground. These conditions led to the removal of 468 buildings.

A further automatic cleaning step involved removing all non-residential buildings, as this thesis focused exclusively on (mixed-)residential buildings. Building use was determined based on the attributes of the BAG building units. At least one of the building uses had to be residential, otherwise the building was discarded. As a result, buildings with an unknown function were also removed. However, these mainly corresponded to small structures such as sheds or larger agricultural buildings and greenhouses, which were not of interest to the analysis. This led to the removal of 98,692 buildings.

The final automatic cleaning steps focused on handling missing feature values. Firstly, any buildings that were missing height information were removed. This is because building height was used in the semi-automatic cleaning process to determine which buildings were incorrectly labelled. Therefore, removing any buildings with missing height data prevented potentially incorrectly labelled buildings from being used as input to the algorithms. This led to the removal of 10,915 buildings. In addition, any other features with missing values were imputed (see Section 3.3.1). For the final model, 689 buildings had at least one missing value, which is approximately 0.4% of the total input data. For these buildings, 12% of all feature values required imputation.

4.3.2 Semi-automatic cleaning

The first part of the semi-automatic cleaning process focused on removing buildings which were labelled incorrectly or had an erroneous extracted height. In order to do this, the strong correlation between building height and the number of floors was utilised. The process consisted of three main steps. The data distribution at the start of the process and after each step is visualised in Figure 4.5 using box plots (left) and violin plots (right).

Box plots and are used to visualise the data distribution in terms of statistical measures. The line in the middle of each box represents the median or 50th percentile. The upper and lower limits of each box represent the 75th and 25th percentiles respectively, meaning that the box length corresponds to the interquartile range. The lines extending from the boxes, called whiskers, correspond to the upper and lower limits of each box plus and minus 1.5 times the interquartile range respectively. Any data points beyond these limits are shown as diamonds. Violin plots show the probability density of the data distribution [Hintze and Nelson, 1998]. These plots were used to show the distribution for slanted and horizontal roofs separately, allowing the influence of roof type to be analysed.

The first step involved removing any buildings with an unrealistic average storey height. This was computed by dividing building height (70th percentile) by the number of floors. Any buildings with an average storey height of less than 1.5 meters were removed, which corresponded to 377 cases. This step was performed in order to place greater emphasis on negative outliers since, from an architectural standpoint, exceptionally small storey heights are less likely than large average storey heights. The resulting data distribution is shown in Figure 4.5c and Figure 4.5d.

The second step involved removing any data points that exceeded the 75th and 25th percentiles plus and minus 1.5 times the interquartile range respectively. This led to the removal of 8425 buildings. These are the outliers shown as diamonds in Figure 4.5c. This step was repeated a second time because some obvious outliers remained, mainly for buildings with 1 floor (visible in Figure 4.5e). This last step led to the removal of a further 2356 buildings. The final distribution is shown in the last row of Figure 4.5. A scatter plot of height versus number of floors is provided in Figure 4.4 to show the buildings kept and removed by the semi-automatic cleaning process. In total, 11,158 buildings were removed.

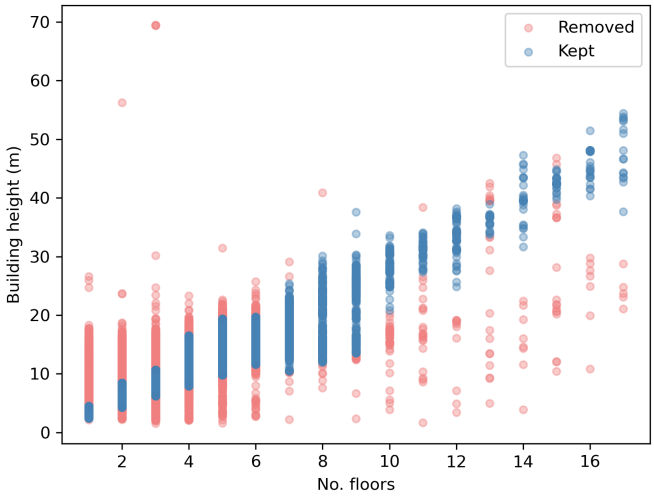
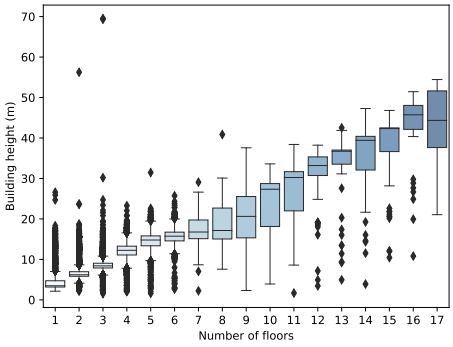


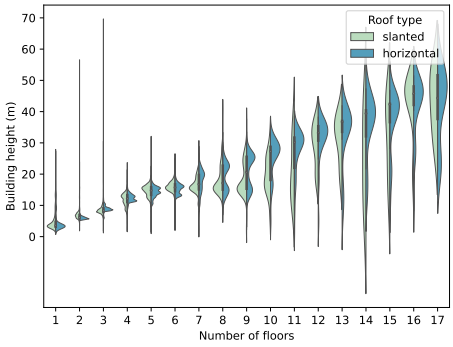
Figure 4.4: Scatter plot of buildings kept and removed by the semi-automatic cleaning steps

The remaining semi-automatic cleaning steps focused on removing clear errors and outliers from the NIA and LOD1.2 volume. As mentioned in Section 2.2.2, the NIA is often incorrectly registered in the BAG. However, since the NIA does not correspond to the total internal floor area, it is difficult to clean automatically. A semi-automatic approach based on trial-and-error was used to determine which cases should be removed. In the end, any cases where the NIA was unrealistically low (less than 10m^2 per floor) were removed, corresponding to 83 buildings.

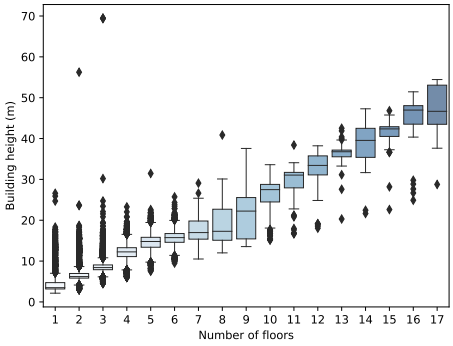
LOD1.2 volume was cleaned by comparing the extracted volume to the volume obtained by multiplying the footprint area with 70th percentile building height. A discrepancy between these values can be caused because underground building sections are included in the BAG footprint. Despite this, some buildings still had an unrealistically low volume. Any cases with an extracted LOD1.2 volume that was smaller than half the volume obtained through multiplication were removed. This corresponded to 153 buildings. In addition, any cases with an extracted volume that was more than twice as large as the volume obtained through multiplication were removed. This corresponded to 98 buildings.



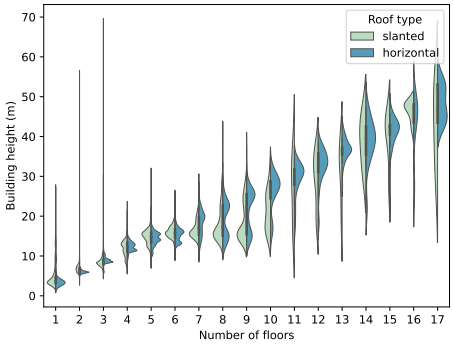
(a) Box plot of input data



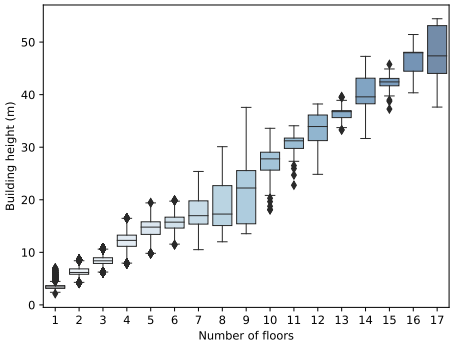
(b) Violin plot of input data



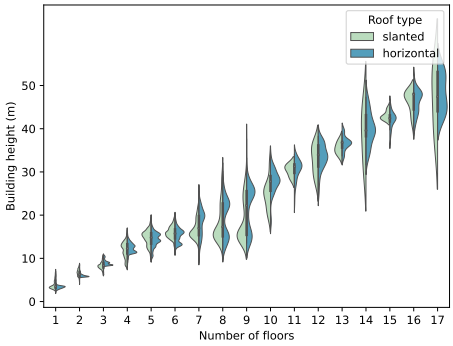
(c) Box plot after step 1



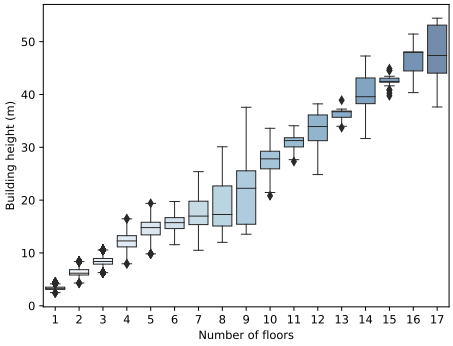
(d) Violin plot after step 1



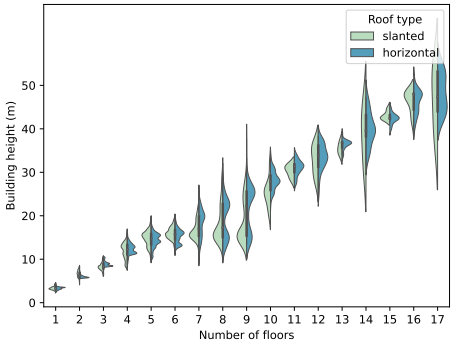
(e) Box plot after step 2



(f) Violin plot after step 2



(g) Box plot after step 3



(h) Violin plot after step 3

Figure 4.5: Semi-automatic data cleaning steps

4.3.3 Manual cleaning

Since the semi-automatic cleaning process relied on using trends in the data distribution to determine incorrectly labelled buildings, it was not performed for buildings with more than 17 floors. This is because there was an insufficient number of buildings to provide reliable trends above 17 floors (see Figure 4.3). Instead, these buildings were cleaned through manual inspection using Google Street View. The manual cleaning process was performed for 118 buildings, of which 53 were found to be incorrectly labelled. The floor count of incorrectly labelled buildings was corrected based on the Street View images. This was challenging because sometimes the number of floors differed depending on the angle inspected or could not be easily determined from the building's exterior. However, attempting to correct the floor count was considered better than completely discarding incorrectly labelled buildings. In cases where the images were not recent enough to validate the number of floors, the original training labels were kept. In order to ensure that the height of buildings above 17 floors was also correct, any buildings with an average storey height of less than 2 meters or more than 4.5 meters were removed. This led to the removal of 27 buildings.

4.4 Computation of 3D geometric features

The extraction of features from the LOD1.2 and LOD2.2 models of the 3D BAG was performed in Python. This is in contrast to the other features, which were extracted in the database directly. The 3D features were handled differently because it was easier to process and visualise the 3D geometries in Python.

Version 21.03.1 of the 3D BAG was used to extract the 3D geometric features. In this version of the dataset, the geometry of each building is reconstructed as a triangular mesh. These meshes are stored in the database as `MultiPolygonZ` geometries. This means that the triangular mesh faces of each building are stored as a collection of 3-dimensional polygons. Each triangular face is associated with a semantic value describing the surface type. In order to extract the roof and wall surface area, the area of the polygons corresponding to roof surfaces and exterior wall surfaces was computed.

The extraction of ridge and eave height from slanted roofs also relied on the semantic values of each polygon. These features were extracted using a three-step process (shown in Figure 4.6). In the first step of this process, step `(a)` in the figure, the maximum z-coordinate of each roof surface polygon was stored. Ridge height was then calculated as the 90th percentile of these coordinates, minus ground height. The 90th percentile was used rather than the maximum in order to take into account data artifacts that had resulted in spikes in some meshes and the occasional presence of chimneys.

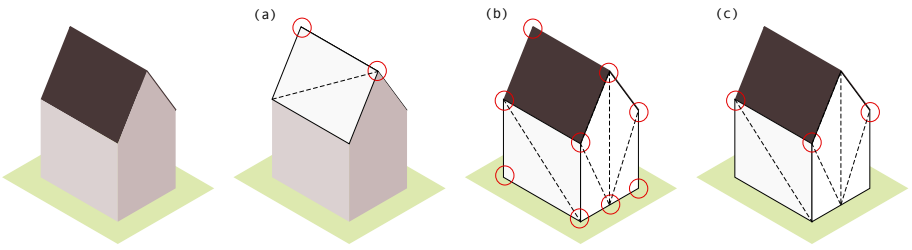


Figure 4.6: Extraction of ridge and eave height from slanted roofs

In order to extract eave height, the z-coordinates of all wall surface polygons were first extracted, as shown in step `(b)`. Then, in step `(c)`, the wall surface coordinates that did not correspond with either the ground height or the maximum roof surface coordinates were extracted. Eave height was calculated as the median of these coordinates, minus ground height. This approach was used rather than determining the minimum z-coordinate of the roof in order to make it more applicable to buildings with lower storey extensions. For these buildings, the minimum z-coordinate would correspond to the roof of the extension, whereas eave height should correspond to the main building. For buildings with

symmetrical roofs, using the median z-coordinate provides reliable results. However, if the roof is unsymmetrical or extensions are present, the median value can be higher or lower than the actual eave height (Figure 4.7)

(a) Unsymmetrical roof shape (eave height higher than reality)

(b) Lower storey extensions (eave height lower than reality)

Figure 4.7: Examples of buildings where eave height extraction is unreliable

For buildings with completely horizontal roofs, ridge height was computed as the 90th percentile of the roof surface z-coordinates, minus ground height. Eave height was set equal to ridge height, meaning that the difference would be zero. For multiple horizontal roofs, an attempt was made to define ridge and eave height in such a way that they could be used to distinguish buildings with elevator shafts. As discussed in Section 3.2.1, these structures increase the height of a building but should not contribute to the floor count. Ridge height was computed as the 90th percentile of the roof surface z-coordinates while eave height was computed as the 75th percentile. A larger difference between these percentiles could indicate that elevator shafts are present. However, the disadvantage of this approach is that the 75th percentile could also correspond to lower roof sections, such as balconies or porches.

Finally, the extraction of volume required the individual polygons to be reconstructed into a mesh. In order to do this, the `PyVista` library was used [Sullivan and Kaszynski, 2019]. This is a Python library for 3D visualisation and mesh analysis. The geometric validity of the reconstructed mesh was evaluated using the `val3dity` library.[2] In total, 0.6% of buildings were found to be invalid in LOD1.2 and 4.3% in LOD2.2. Geometrically invalid meshes were voxelised using `PyVista`. Voxel size was computed based on the number of voxels chosen to fit the length of the mesh. Increasing the number of voxels leads to a higher grid resolution [Mulder, 2015].

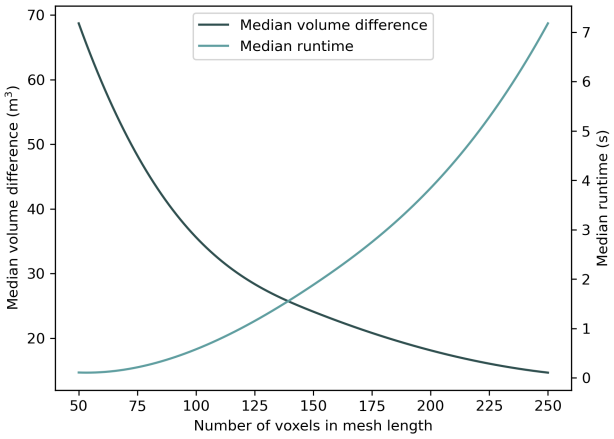
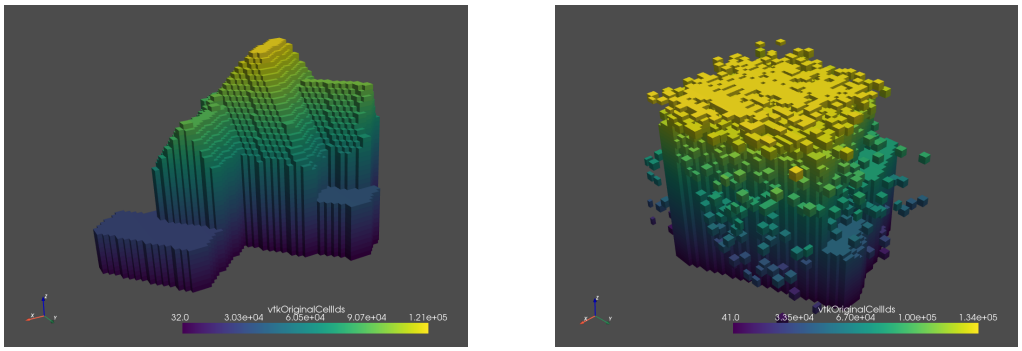


Figure 4.8: Effect of number of voxels on the median runtime per building and the median volume difference between the original mesh and voxelised mesh per building (in LOD2.2)

[2] https://github.com/tudelft3d/val3dity/

The optimal number of voxels was investigated by computing the voxelised volume of geometrically valid buildings at different grid resolutions. The difference between the voxelised volume and the mesh volume was computed for each resolution, as well as the time taken to voxelise each building. This analysis was performed for both levels of detail. Figure 4.8 shows the effect of the number of voxels on the median volume difference and median runtime per building in LOD2.2. Based on this, a value of 90 voxels was selected for LOD2.2. A similar analysis was performed for LOD1.2 and a value of 75 voxels was selected. These values were chosen to keep the median runtime per building below one second, allowing the overall processing time to remain feasible for all invalid buildings.

After voxelisation, building volume was computed from the voxels. This volume was compared to the volume of the convex hull in order to identify cases that had been incorrectly voxelised. Examples of correctly and incorrectly voxelised meshes are provided in Figure 4.9. The convex hull of a building should either be larger than or equal to the building geometry. Therefore, if the voxelised volume was 10% larger than the convex hull, the convex hull volume was used instead. A value of 10% was chosen based on trial-and-error to identify as many incorrectly voxelised meshes as possible.



(a) Correctly voxelised (b) Incorrectly voxelised

Figure 4.9: Examples of voxelised meshes

4.5 Feature selection approaches

In order to rank the input variables according to their predictive power, different feature selection approaches can be used. These can be subdivided into *filter*, *wrapper* and *embedded* methods [Bousquet et al., 2004; Chandrashekar and Sahin, 2014; Guyon and Elisseeff, 2003].

1. Filter methods use statistical measures to rank the input features based on their relationship to the target variable. The highest ranking features are then selected, using either a threshold value or the *k-best* features. The benefit of this approach is that it is computationally light and independent of the machine learning algorithm used [Chandrashekar and Sahin, 2014]. This means that the likelihood of overfitting is reduced, as the selection is not tuned for a specific learning algorithm [Guyon and Elisseeff, 2003]. However, since univariate statistical tests are performed, the influence of other features is not considered during the process [Alamdari, 2006]. As a result, features that have low individual relevance but a high relevance when combined with other features are still discarded. Furthermore, if the features are not independent, the feature subset may contain many correlated features, leading to redundancy [Chandrashekar and Sahin, 2014; Duch, 2006].

2. Embedded methods are integrated as part of the training process, meaning they are dependent on the machine learning algorithm used. The importance of each feature is derived based on its contribution to the predictive model. Since this is determined during training, the method does not become too computationally complex [Chandrashekar and Sahin, 2014; Guyon and Elisseeff, 2003]. A further benefit is that the interaction between features is taken into account, enabling a better understanding of the training dataset to be obtained. A common example of an embedded method is the impurity-based

feature importance built into tree-based models [Lal et al., 2006]. For algorithms that assign weights to each feature, the model coefficients can be used to rank the feature importances. Features that receive a higher weight have a larger influence on the model predictions and are considered more important [Chandrashekar and Sahin, 2014]. The drawback is that this approach is only applicable to certain algorithms. Furthermore, feature importances can be influenced by correlation bias. Features belonging to larger groups of correlated features have been found to receive smaller weights due to their shared responsibility in the model [Toloşi and Lengauer, 2011].

3. Wrapper methods use model performance to assess the relative usefulness of subsets of variables. The model is considered a black-box and is trained multiple times on different subsets of features. Model feedback is generally obtained through cross-validation [Guyon and Elisseeff, 2003]. As a result, wrapper methods are often computationally complex. Furthermore, they can be prone to overfitting, since they are highly dependent on the training data and algorithm used [Chandrashekar and Sahin, 2014].

Due to the limitations of wrapper methods, this thesis focused solely on filter and embedded methods. Three approaches were used, as discussed in further detail in the following sections. Each approach was developed to reduce the amount of input data by at least half (from 25 features to 10 or 12).

4.5.1 Filter-based

The filter-based approach can be implemented using a variety of statistical measures. In this thesis, the Pearson correlation coefficient and Mutual Information (MI) were considered. The Pearson coefficient captures linear relationships [Chandrashekar and Sahin, 2014]. It ranges from -1 to 1, where -1 indicates a strong negative correlation, 1 a strong positive correlation and 0 no correlation. MI measures the dependency between variables, allowing it to also capture non-linear relationships. A value of 0 means that two variables are independent and higher values mean higher dependency [Chandrashekar and Sahin, 2014]. The influence of both linear and non-linear relationships were considered because two of the algorithms used in this thesis (RF and GB) are non-parametric.

Table 4.3: Top 10 features based on their correlation to the number of floors

	Mutual Information		Pearson Correlation	
	Feature	**Value**	**Feature**	**Value**
1	Height (70th)	1.0	Height (70th)	1.0
2	Height (max)	0.89	Height (max)	0.79
3	Height (50th)	0.85	Height (50th)	0.58
4	Roof area (LOD1.2)	0.63	Roof area (LOD1.2)	0.11
5	Roof area (LOD2.2)	0.60	Roof area (LOD2.2)	0.11
6	Net internal area	0.59	No. units	0.05
7	Volume (LOD1.2)	0.51	Volume (LOD1.2)	0.04
8	Volume (LOD2.2)	0.50	Volume (LOD2.2)	0.04
9	Population density	0.47	Net internal area	0.03
10	% multi-household	0.34	% multi-household	0.03

The top ten features based on MI and Pearson's correlation coefficient are shown in Table 4.3. The two ranking methods provide very similar results. The top five features are the same in both cases and the results differ by only one feature overall. MI places more importance on NIA and selects population density rather than the number of units. The MI scores are also higher, suggesting that this statistical measure is able to capture more complex relationships present in the data.

Since NIA appears to have a stronger relationship to the number of floors than the number of units, the features selected based on MI were used. The relationship of these features to the number of floors is visualised in Figure 4.10. As expected, building height has the strongest level of correlation, with 70th percentile height ranking highest. Two other 3D geometric features, roof area and volume, are also

found to be related to the number of floors. The correlation is approximately the same irrespective of the level of detail used. The other highest ranking features are NIA, population density and the percentage of multi-household buildings. Since the area-based approach presented in Section 2.2.2 uses NIA to estimate the number of floors, it makes sense that it ranks quite highly. The presence of two statistical features is more surprising, as these were defined at a neighbourhood level rather than per building.

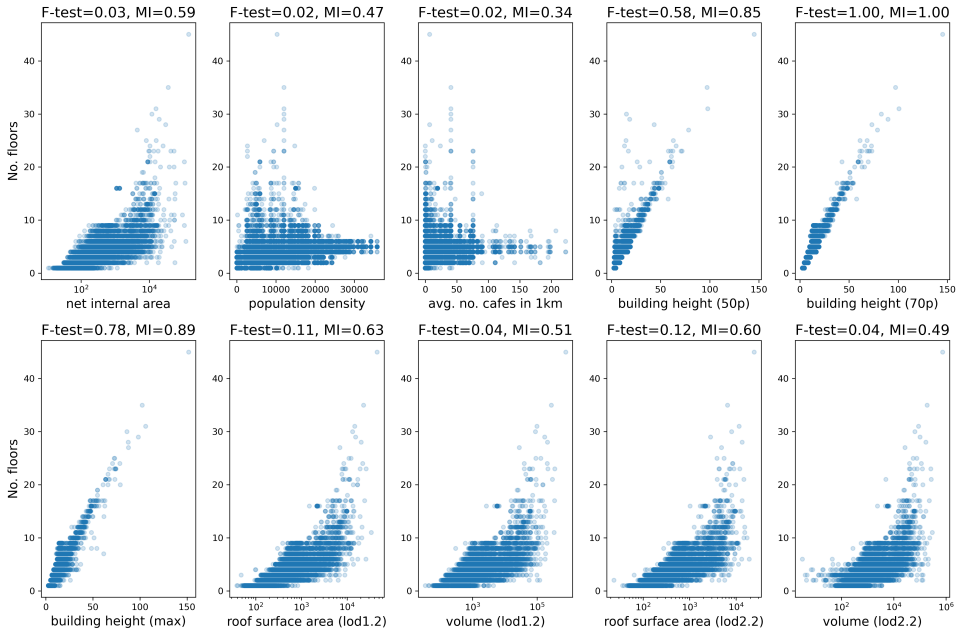


Figure 4.10: Top 10 features based on Mutual Information plotted against the number of floors

4.5.2 Embedded

Embedded feature selection was implemented using the impurity-based importance of the tree-based models (RF and GB) and the weights of the linear SVR model. The ten most important features obtained for each algorithm are summarised in Table 4.4, alongside the importance/weight assigned to each feature. The results are quite varied, highlighting the dependence on the algorithm used. However, all three algorithms consider 70th percentile building height to be the most important feature. Notably, this feature was also ranked highest by the filter-based method.

The RF and GB algorithms consider many of the same features to be important, although rank them slightly differently. These are mainly 3D geometric or cadastral features, but a number of statistical features are also considered to be important. This makes the results relatively similar to those obtained by the filter-based method. However, fewer 3D geometric features are selected. Furthermore, unlike the filter-based method, the LOD2.2 features rank higher than their LOD1.2 equivalents, which are not included in the feature subsets. It is also interesting to note that 70th percentile building height has a much higher importance than all other features. However, since the importance scores are based on the training set, the other features could still be useful for predicting the number of floors of unseen cases.

The results obtained for the SVR model are quite different from the other two algorithms. The features with the highest weight are mainly categorical (e.g roof shape and building type). Notably the highest ranking features for RF and GB do not include any categorical features. After one-hot encoding, the categorical features are binary, meaning they are less likely to obtain a high impurity-based importance score. This is because impurity importance is biased towards high cardinality features (see Section 3.4.) However, for SVR it seems that the ability of embedded methods to rank both categorical and numerical features is a further advantage compared to filter-based methods. Filter-based approaches can only consider numerical features, since it is not possible to compute statistical measures for text-based features,

even after they have been one-hot encoded. Aside from categorical features, the SVR model gives the highest weight to building height and volume. In contrast to the other two algorithms, LOD1.2 ranks higher than LOD2.2. Furthermore, maximum building height is not included, unlike all previously discussed feature subsets. Instead, 50th percentile height receives more weight.

Table 4.4: Top 10 features based on model importance

	Random Forest		Gradient Boosting		Support Vector Regression	
	Feature	**Value**	**Feature**	**Value**	**Feature**	**Value**
1	Height (70th)	0.860	Height (70th)	0.883	Height (70th)	1.18
2	Net internal area	0.025	Height (max)	0.071	Slanted roof	0.70
3	Height (max)	0.024	Net internal area	0.020	Single horizontal roof	0.63
4	Construction year	0.014	No. units	0.005	Multi-horizontal roof	0.62
5	Height (50th)	0.008	Height (50th)	0.004	Apartment	0.55
6	Avg. no. cafes in 1km	0.008	Construction year	0.004	Terraced	0.53
7	Population density	0.007	Roof area (LOD2.2)	0.003	Semi-detached	0.52
8	Ridge - eave height	0.006	Avg. no. cafes in 1km	0.002	Detached	0.35
9	Volume (LOD2.2)	0.005	% multi-household	0.001	Volume (LOD1.2)	0.24
10	No. neighbours in 100m	0.005	Ridge - eave height	0.001	Height (50th)	0.18

Overall, the results of the filter and embedded methods are quite varied, highlighting the difficulty in selecting the best subset. However, a number of features were not included in any of the subsets, suggesting they are not relevant to the prediction problem. These features were wall surface area, building function and all 2D geometric features, aside from the number of neighbours.

4.5.3 Multicollinearity reduction

A drawback of filter and embedded methods is that the selected subsets include many similar features. This is because the input variables were not independent. As a result, there may be a high level of correlation between features. Multicollinearity is particularly problematic for algorithms based on linear regression, as the variables are assumed to be independent and uncorrelated [Hill and Adkins, 2007]. Since the SVR algorithm used in this thesis is based on a linear kernel, an alternative feature selection method was developed to reduce multicollinearity.

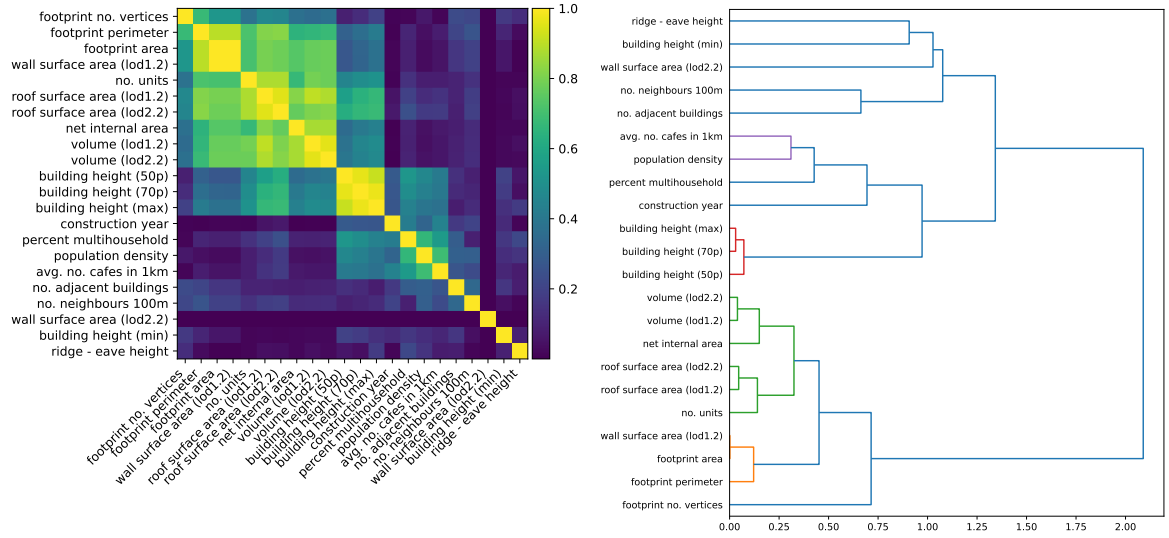
The level of correlation between the input variables can be measured using the Variance Inflation Factor (VIF). This describes the extent to which the variance of an independent variable is increased by its correlation to other variables. A value of 1 indicates the absence of collinearity and, as a general rule of thumb, values above 5 or 10 indicate high collinearity [James et al., 2021]. The VIF scores of the filter-based subset and the subsets based on feature importance are shown in Table 4.5. The subset based on SVR weights was not included as VIF scores cannot be computed for categorical features.

The VIF scores show that there is a high level of multicollinearity present in the selected feature subsets, particularly for the filter-based subset. Seven of the features selected using the filter-based method have a VIF score higher than 5. The subset based on GB feature importance performs slightly better, as the number of features with a VIF score higher than 5 is reduced to five. Finally, the subset based on RF feature importance performs best, with only three VIF scores higher than 5. However, these three features have very high scores, indicating they are highly correlated with each other.

The correlation between the input features is also visualised with a correlation matrix (Figure 4.11a). This matrix has been sorted to place correlated features close together, resulting in clusters of yellow and green. As expected, the clusters of correlated features are mostly formed by sets of similar features. For instance, the different representations of building height are clustered together, as well as many of the 3D geometric features. Since many of the features selected using the filter-based approach correspond to these clusters, it makes sense that this subset has the highest VIF scores.

Table 4.5: Variance inflation factor of feature subsets

	Filter-based (MI)		RF feature importance		GB feature importance	
	Feature	**VIF**	**Feature**	**VIF**	**Feature**	**VIF**
1	Height (70th)	51.8	Height (70th)	44.9	Height (70th)	44.9
2	Roof area (LOD1.2)	38.6	Height (max)	26.4	Height (max)	27.6
3	Volume (LOD1.2)	26.5	Height (50th)	18.0	Height (50th)	18.2
4	Height (50th)	23.0	Volume (LOD2.2)	4.3	Roof area (LOD2.2)	9.2
5	Roof area (LOD2.2)	21.6	Net internal area	4.1	No. units	5.1
6	Height (max)	20.0	Population density	2.4	Net internal area	2.4
7	Volume (LOD2.2)	17.0	Avg. no. cafes in 1km	2.3	% multi-household	1.9
8	Net internal area	4.7	Ridge - eave height	1.6	Ridge - eave height	1.8
9	% multi-household	2.0	Construction year	1.3	Avg. no. cafes in 1km	1.6
10	Population density	1.9	No. neighbours in 100m	1.2	Construction year	1.4



(a) Correlation matrix

(b) Hierarchical clustering

Figure 4.11: Relationship between input features based on the Pearson correlation coefficient

The third feature selection approach focused on reducing the groups of correlated features. This approach was based on *hierarchical clustering*, a process which enables closely related variables to be grouped together based on a similarity measure [Rokach, 2010]. In this case, Ward's linkage was used. This function computes the "distance" between two clusters as the increase in the Error Sum of Squares (ESS) after merging two clusters together. The function aims to minimise the increase of ESS at each step [Ward, 1963]. The resulting clusters are visualised by the dendrogram shown in Figure 4.11b. A threshold distance of 0.4 was used to assign features to the same cluster. This value was chosen in order to group together as many similar features as possible. The groups shown in red, green, orange and purple represent different clusters, while the features linked in blue were not assigned to a cluster.

In order to select a feature subset with reduced multicollinearity, a filter-based approach was firstly performed on each cluster. The feature with the highest MI score was selected per cluster. Then, the features with the ten highest MI scores were selected from the best feature per cluster and the remaining unclustered features. The results are shown in table Table 4.6, alongside the MI scores. Since the embedded method for SVR consisted of many categorical features, these features were also added to the subset. It is noticeable that the MI score of some features is quite low. However, a number of these features were

also considered important by the embedded methods based on RF and GB. Features with low individual relevance can still be useful when combined with other features [Guyon and Elisseeff, 2003]. For comparison, the VIF scores are also shown. All scores are lower than 10 and only one feature has a score slightly higher than 5, showing that multicollinearity was successfully reduced.

Table 4.6: Feature subset with reduced multicollinearity

	Feature	MI score	VIF score
1	Building height (70th)	1.00	3.28
2	Roof area (LOD2.2)	0.60	6.63
3	Population density	0.47	2.16
4	% multi-household	0.35	2.30
5	Construction year	0.24	1.31
6	Footprint perimeter	0.18	4.34
7	Building height (min)	0.14	1.09
8	Ridge - eave height	0.12	1.17
9	No. adjacent buildings	0.05	1.29
10	No. neighbours in 100m	0.04	1.42
11	Roof type	-	-
12	Building type	-	-

The algorithms used in this thesis were trained on the feature subsets obtained for the three feature selection approaches presented in this section. This led to nine different models (i.e. three per algorithm). The performance of each of these models is compared in Section 5.1.1. Based on this, the best predictive model per algorithm is selected.

4.6 Hyperparameter tuning results

The hyperparameters of the best model per algorithm were tuned using a randomised grid search over 75 different parameter combinations (as discussed in Section 3.3.2). In order to determine appropriate parameter values to test, validation curves were plotted. These plots show the influence of each hyperparameter on model performance. The validation curves of the GB and SVR hyperparameters are shown in Figure 4.12 and Figure 4.13 respectively. The plots for RF are provided in Figure C.1 of the Appendix. To obtain these plots, each hyperparameter was altered in isolation and 5-fold cross-validation was used to evaluate model performance. Since the dependence between hyperparameters is not considered, these plots are not fully representative of the impact on model performance. However, they are still useful for gaining an initial understanding of which values to test.

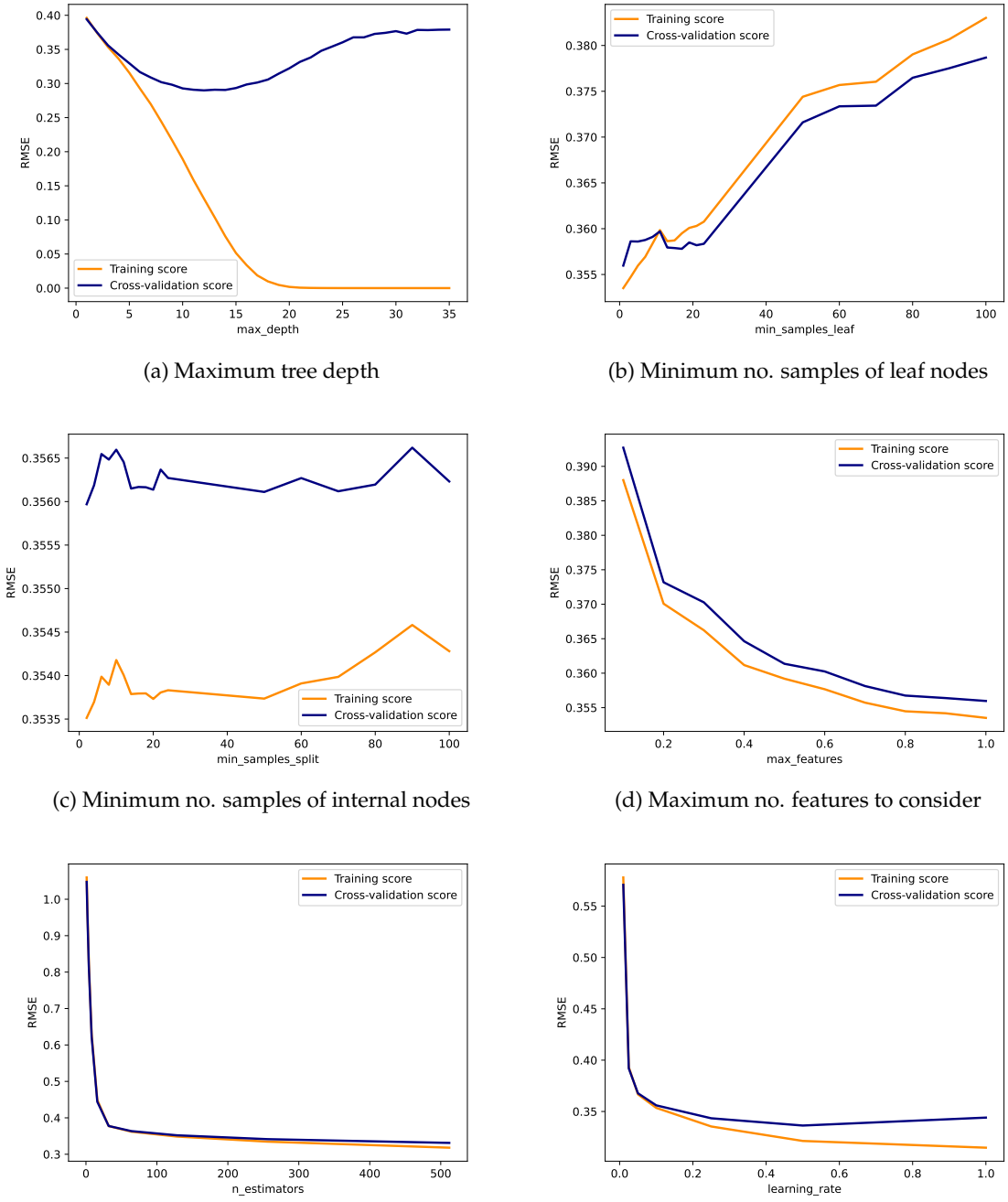
4.6.1 Tree-based algorithms

The hyperparameters of the RF and GB algorithms can be sub-divided into tree-specific parameters and ensemble parameters. The tree-specific parameters are the same for both algorithms. The impact of each hyperparameter on model performance was assessed using the RMSE. This error metric was chosen because of its higher sensitivity to larger errors (as discussed in Section 3.3.2).

In `scikit-learn`, the maximum depth of each tree in the ensemble is controlled by the `max_depth` parameter. Figure 4.12a shows that increasing the tree depth of the GB algorithm initially led to a reduction in RMSE for both the training and cross-validation sets. However, beyond a certain depth, the cross-validation error no longer improved and even began to increase. At the same time, the training error continued to decrease until a plateau. This shows that increasing the maximum tree depth caused the

model to overfit the training set and prevented it from generalising to the test set. A similar pattern was observed for RF, although the level of overfitting was lower for larger tree depths.

The minimum number of samples required in leaf nodes and internal nodes is controlled by the parameters: `min_samples_leaf` and `min_samples_split`. Increasing these parameters causes the trees to become more constrained because more samples are required before a node can be created. The effect of altering these parameters is shown in Figure 4.12b and Figure 4.12c respectively. These plots show that increasing both parameters led to a larger RMSE, indicating that the GB model increasingly underfit the data. A similar pattern was observed for RF.



(a) Maximum tree depth

(b) Minimum no. samples of leaf nodes

(c) Minimum no. samples of internal nodes

(d) Maximum no. features to consider

(e) No. trees in ensemble

(f) Learning rate

Figure 4.12: Validation curves for Gradient Boosting hyperparameters

The final tree-specific parameter that was tuned was the `max_features`. This parameter controls the maximum number of features to consider when looking for the best split. Figure 4.12d shows that increasing this parameter led to a decrease in the RMSE. The x-axis of this plot shows the fraction of features considered at each node, meaning that a value of 1 corresponds to all features. For GB, using all features resulted in the lowest RMSE. However, for RF, the RMSE reached a plateau beyond a certain fraction of features, suggesting that it was not necessary for all features to be considered at each node.

The GB ensemble is controlled by two main parameters. Firstly, the number of trees is determined by the `n_estimators` parameter. Increasing the number of trees generally leads to better model performance, however using many trees also slows down the training process. Figure 4.12e shows that the RMSE was reduced by increasing the number of trees. The decrease in RMSE was initially very high, since using only a few trees causes the algorithm to underfit the training data. Beyond a certain point, the decrease was more gradual, showing that using more trees does not substantially improve model performance. The contribution of each tree is determined by the `learning_rate` parameter. A lower learning rate means that more trees are required to fit the data, but generally results in better model generalisation [Géron, 2019]. Figure 4.12f shows that increasing the learning rate initially led to a reduction in RMSE. However, when the learning rate became higher, the gap between the two curves increased. This shows that the model started to overfit the training data and did not generalise well to the validation set.

Similar to GB, the number of trees in the RF ensemble is controlled by the `n_estimators` parameter. Increasing the number of trees in the RF ensemble also led to better model performance. However, beyond a certain beyond a certain number of trees, the reduction in RMSE also reached a plateau. This shows that the number of trees should not be unnecessarily high, otherwise the training time will be increased without a noticeable improvement in model performance. The RF ensemble is also controlled by a second hyperparameter: `bootstrap`. This was not plotted as a validation curve as it can either be set to `True` or `False`. This parameter determines whether samples are drawn with replacement or not (see Section 2.3.2).

Table 4.7: Overview of Random Forest hyperparameters tested and selected

Hyperparameter	**Values tested**			**Value selected**
	Start	**End**	**Step**	
`max_depth`	10	42	4	22
`min_samples_split`	2	50	4	14
`min_samples_leaf`	1	45	5	1
`max_features`	5	10	1	5
`n_estimators`	50	350	25	300
`bootstrap`	True / False			False

Table 4.8: Overview of Gradient Boosting hyperparameters tested and selected

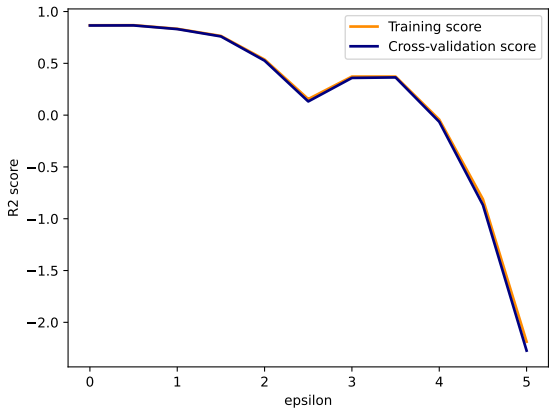
Hyperparameter	**Values tested**			**Value selected**
	Start	**End**	**Step**	
`max_depth`	2	24	2	16
`min_samples_split`	2	50	4	22
`min_samples_leaf`	1	45	5	1
`max_features`	5	10	1	5
`n_estimators`	150	850	50	550
`learning_rate`	0.01	0.1	0.01	0.02

Table 4.7 and Table 4.8 show the hyperparameter values tested and selected for the RF and GB algorithms respectively. The maximum depth and number of trees of the GB algorithm were tuned using slightly different values from RF. This is because the trees in a GBRT should be less deep so that the ensemble is formed by weak learners (see Section 2.3.3). Since the trees in the GB ensemble are less deep, a higher number of trees was considered to allow the weak learners to fit the data. Therefore, the tuned RF algorithm had a higher maximum depth but fewer trees.

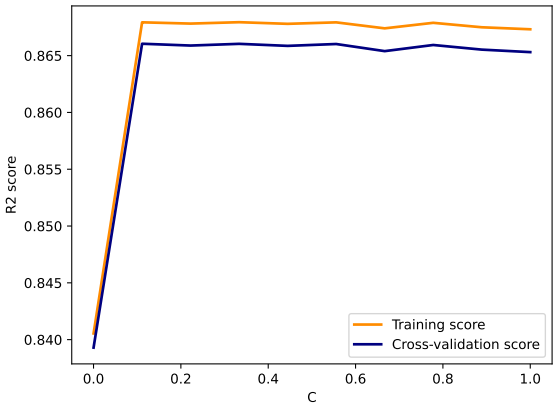
4.6.2 Support Vector Regression

The linear SVR algorithm implemented by the `scikit-learn` library is controlled by six main parameters. Unlike the RF and GB algorithms, the impact of these hyperparameters on model performance was assessed using the coefficient of determination (R^2). This is because RMSE was incompatible with some of the parameter combinations tested. The coefficient of determination provides an indication of the goodness of fit and a higher score indicates better model performance.

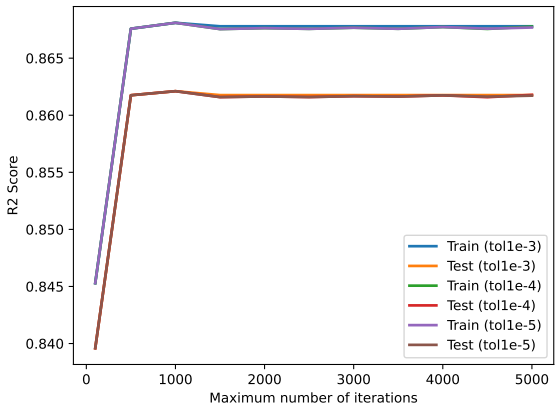
The first SVR hyperparameter, *epsilon*, defines the margin of tolerance for which errors are not penalised (see Section 2.3.4). Figure 4.13a shows that increasing epsilon led to a reduction in model performance for both the training and cross-validation sets, as the R^2 score decreased. A value of 0 can lead to overfitting. However, since the value of epsilon is based on the training set, it is often better to choose a smaller value in order to enable the model to generalise to new data.



(a) Epsilon



(b) Regularisation parameter C



(c) Tolerance and max. no. iterations

Figure 4.13: Validation curves for Linear Support Vector Regression hyperparameters

The *regularisation parameter* (C) determines the penalties assigned to training instances outside the ϵ-boundaries. The level of regularisation is inversely proportional to C, meaning that overfitting occurs if the value is set too high. Figure 4.13b shows that increasing C initially improved model performance for both the training and cross-validation sets. However, the improvement reached a plateau and gradually began to decrease. In addition, once the plateau was reached, the gap between the training and cross-validation scores increased, indicating overfitting.

The *tolerance* and *maximum number of iterations* were evaluated together because these parameters both affect model convergence. Figure 4.13c shows that model performance was not affected by changing the tolerance value. This may be because the linear SVR algorithm was too simple for the prediction

problem, causing the model to underfit the data and provide the same performance score regardless of the values tested. Increasing the number of iterations led to an initial increase in model performance, although a plateau was quickly reached.

The last two SVR hyperparameters were not plotted as validation curves. The *loss* parameter determines the loss function used, which is either ϵ-insensitive or squared ϵ-insensitive. The *dual* parameter determines whether the dual or primal optimisation problem is solved. The primal optimisation problem is preferred when the number of samples is higher than the number of features. The hyperparameters tested and selected for the SVR algorithm are outlined in Table 4.9.

Table 4.9: Overview of Linear Support Vector Regression hyperparameters tested and selected

Hyperparameter	**Values tested**	**Value selected**
`epsilon`	[0.0, 0.01, 0.05, 0.1, 0.5, 1]	0.0
`C`	[0.1 0.3 0.5 0.7 0.9 1.1]	0.9
`tol`	[1e-3, 1e-4, 1e-5]	1e-3
`max_iter`	`start`=1000, `stop`=5000, `step`=500	4500
`loss`	ϵ-insensitive / squared ϵ-insensitive	squared ϵ-insensitive
`dual`	True / False	True

5 Results and analysis

This chapter presents the results and analysis. Firstly, the performance of each model is evaluated in order to determine the quality of the predictions. This evaluation is performed on the models obtained before and after tuning the hyperparameters. Then, a more in-depth error analysis is conducted, with a focus on the model that provides the best predictions. After this, the impact of rounding is investigated in order to determine to what extent this affects the results. In addition, the feature contributions are analysed and the effect of data availability is considered. The predictions are then compared to the geometric approach to establish whether machine learning provides better results. Finally, the applicability of the model to other areas of the Netherlands is explored.

5.1 Model performance

5.1.1 Before hyperparameter tuning

In the first stage of the modelling process, four models were obtained for each algorithm: one based on all features and one for each of the three feature selection approaches presented in Section 4.5. In order to evaluate the performance of these models, the MAE, RMSE, maximum error and accuracy were computed on the predictions obtained for the training and test sets. The results are summarised in Table 5.1. In this table, model 1 corresponds to the filter-based approach, model 2 to the embedded approach and model 3 to the alternative approach that aimed to reduce multicollinearity. The definitions of MAE, RMSE, maximum error and accuracy are provided in Section 3.3.2.

Table 5.1: Model evaluation results before tuning

		MAE		RMSE		Max. error		Accuracy (%)		Training
		Train	Test	Train	Test	Train	Test	Train	Test	time (s)
RFR	All	0.01	0.07	0.08	0.30	5	4	99.5	93.1	243.84
	1	0.01	0.08	0.08	0.31	5	4	99.4	92.4	128.24
	2	0.01	0.07	0.08	0.30	5	4	99.5	93.0	106.26
	3	0.01	0.08	0.08	0.31	4	5	99.4	92.2	99.99
GBR	All	0.13	0.13	0.39	0.40	4	4	87.7	87.6	71.15
	1	0.13	0.14	0.40	0.41	4	5	87.6	87.4	40.71
	2	0.13	0.13	0.39	0.40	4	5	87.7	87.7	31.99
	3	0.14	0.14	0.41	0.41	4	4	86.5	86.5	35.08
SVR	All	0.16	0.16	0.45	0.45	13	6	85.0	85.2	30.90
	1	0.17	0.17	0.46	0.45	18	6	84.1	84.1	24.90
	2	0.18	0.18	0.47	0.47	13	7	83.1	83.1	17.63
	3	0.17	0.17	0.44	0.44	10	4	84.6	84.6	25.65

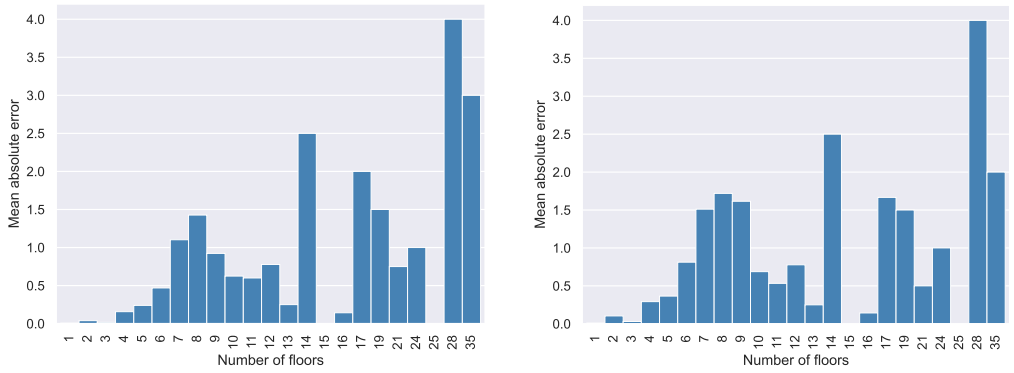
Overall, the models obtained using RF performed best, followed by GB and SVR. However, it is noticeable that RF overfits the training data, as the models perform better on the training set than the test set. The other algorithms have similar results for the training and test sets, suggesting that they do not suffer

from overfitting. This is confirmed by the learning curves provided in Figure C.2 of the Appendix for the models based on all features. These plots show how the training and cross-validation errors are influenced by the amount of data the model is trained on. A gap between the training and cross-validation curves indicates that the model overfits the training data. The only model with a gap between the curves is the one based on RF.

Table 5.1 also shows that the different feature subsets did not have a substantial impact on model performance. The results obtained are almost the same, regardless of whether all features are used or one of the feature subsets. The only noticeable difference occurs for the maximum error of the models based on SVR, which is lowest for the third model. As discussed in Section 3.2.3, feature selection allows less useful features to be removed. This reduces the computational cost of training but also reduces the noisiness of the training data, which can help to improve model performance. However, in this case, feature selection does not appear to improve model performance. Nonetheless, reducing the number of features did not substantially lower the quality of the predictions and helped to reduce training time.

For RF and GB, the results of the second feature selection approach are most similar to the baseline model obtained using all features. This makes sense because this subset was based on the features that these algorithms considered most important. For SVR, the results of the third feature selection approach appear to be the best. The RMSE and maximum error are lowest for this model and better than the results obtained using all features. The MAE and accuracy score are also closest to the baseline model. It makes sense that this approach was more suitable for SVR because this algorithm performs better when multicollinearity is reduced. It is interesting to note that the models obtained using SVR always have a higher maximum error for the training set than the test set. This is because the training set contained one building of 45 floors which had a large prediction error for this algorithm, while a building with a similar number of floors was not present in the test set.

The results seem to indicate that all three models provide very accurate predictions. The best model has an accuracy of 93.1% on the test set and the worst model is still over 80% accurate. However, these results are highly influenced by data imbalance. As discussed in Section 4.1, the dataset is heavily skewed towards lower storey buildings, with 90% of the data below 5 floors. This causes model performance to appear misleadingly good because the models provide accurate predictions for low storey buildings. This is shown by the low MAE obtained for buildings below 5–6 floors in Figure 5.1 and Figure C.3. For this reason, the same error metrics were also computed separately for buildings above 5 floors and with 5 floors or less. The results are summarised in Table 5.2. These results are based on the test set, since this provides the best indication of model generalisation.



(a) Random Forest (all features)

(b) Gradient Boosting (all features)

Figure 5.1: Mean absolute error per number of floors for the test set

Model performance is clearly very different when a distinction is made based on the number of floors. The results obtained for buildings with 5 floors or less correspond with the results shown in the previous table. However, for buildings above 5 floors, the quality of the predictions is much lower. For instance, RF is almost 95% accurate below 5 floors, but only 50% accurate above. This difference is even more extreme for GB and SVR. Overall, RF performs better than the other two algorithms for both lower and

higher storey buildings. This can be seen in all four error metrics, apart from the maximum error. The maximum error below 5 floors is slightly lower for the models obtained using GB and SVR.

Table 5.2: Model evaluation results above and below 5 floors for the test set before tuning

		MAE		RMSE		Max. error		Accuracy (%)	
		> 5	≤ 5	> 5	≤ 5	> 5	≤ 5	> 5	≤ 5
RFR	**All**	0.64	0.06	1.02	0.24	4	3	50.9	94.4
	1	0.68	0.06	1.08	0.26	4	3	49.4	93.7
	2	0.65	0.06	1.02	0.24	4	3	50.3	94.4
	3	0.67	0.07	1.05	0.26	5	3	49.8	93.5
GBR	**All**	0.98	0.11	1.29	0.33	4	2	25.6	89.6
	1	1.04	0.11	1.36	0.33	5	2	24.1	89.4
	2	0.98	0.11	1.31	0.33	5	2	27.0	89.6
	3	1.02	0.12	1.33	0.35	4	2	23.7	88.4
SVR	**All**	1.17	0.13	1.50	0.36	6	5	20.3	87.2
	1	1.07	0.14	1.45	0.38	6	2	26.9	85.9
	2	1.18	0.15	1.53	0.39	7	2	20.1	85.1
	3	1.14	0.13	1.46	0.37	4	2	21.1	86.7

Based on the model evaluation results, the best model per algorithm was selected. For RF, the model obtained using the embedded feature selection approach was chosen (model 2). This model provided almost the same results as the model trained on all features and performed best for all four error metrics, both above and below 5 floors. For the GB algorithm, model 2 was also selected. Out of the models based on feature subsets, this model had the lowest MAE and RMSE both above and below 5 floors. It also had the highest accuracy above 5 floors and was even slightly more accurate than the model based on all features. Apart from this, model 2 performed almost exactly the same as the baseline model, except that the maximum error above 5 floors was one floor higher and RMSE was also slightly higher.

For SVR, the model obtained using the alternative feature selection approach was chosen (model 3). This model did not perform best for all error metrics, but seemed the best choice overall. Model 1 performed better above 5 floors in terms of MAE, RMSE and accuracy. However, model 3 performed best for these metrics below 5 floors. It also had the lowest maximum error above 5 floors. Furthermore, since linear SVR is negatively affected by multicollinearity, model 3 seemed to be the more appropriate choice.

5.1.2 After hyperparameter tuning

After tuning the hyperparameters of the best model per algorithm, the same model performance evaluation was performed. The results for each model are compared before and after tuning in Table 5.3. The performance of the GB algorithm improved most after tuning, resulting in an accuracy of almost 94.5% below 5 floors and 52.3% above. The MAE below 5 floors was almost halved after tuning and it was reduced by a third above 5 floors. The maximum error above 5 floors was reduced by one floor, but below 5 floors it increased.

RF showed very little improvement after tuning. The MAE, RMSE and accuracy even became slightly worse for buildings above 5 floors. This could be because the RF algorithm was more sensitive to data imbalance. As discussed in Section 4.1, most algorithms focus on minimising the overall error rate. Since the majority of buildings were below 5 floors, the tuning process focused on improving the results for these buildings. In the case of RF, this caused model performance to decline for buildings above 5 floors. In addition, since the accuracy achieved for buildings below 5 floors was already quite high, the tuning process only resulted in a slight improvement in model performance. This mainly occurred for the maximum error, which was reduced by one floor.

SVR showed a small amount of improvement after tuning. Mainly the MAE, RMSE and accuracy above 5 floors improved. However, the maximum error did not improve at all and the other error metrics became slightly worse below 5 floors. After tuning, the overall accuracy of the model based on SVR was still quite low, indicating that this is the least suitable algorithm for the prediction problem. The performance of the tuned GB model was almost the same as the best model obtained using RF. However, the GB model provided a slightly better MAE, RMSE and accuracy above 5 floors. For this reason, the tuned GB model seems to be the best predictive model. This is further analysed in the following section.

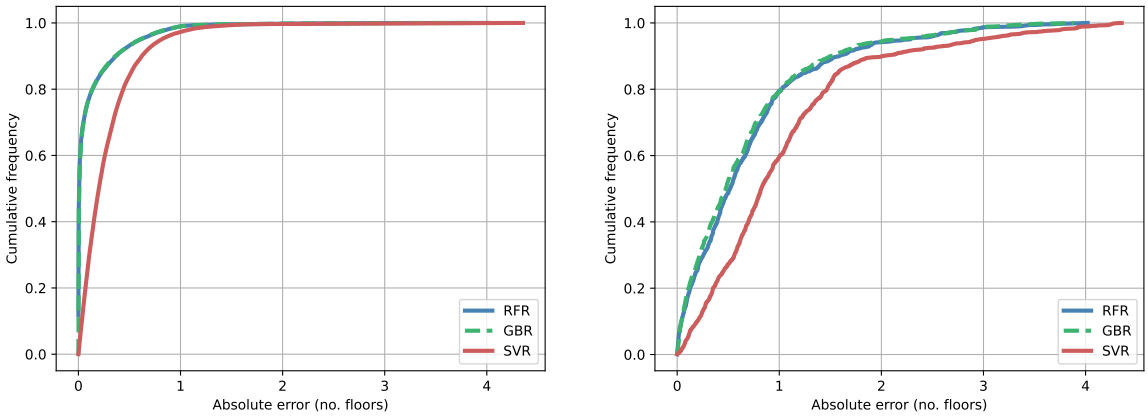
Table 5.3: Model evaluation results above and below 5 floors for the test set after tuning

		MAE		RMSE		Max. error		Accuracy (%)	
		> 5	≤ 5	> 5	≤ 5	> 5	≤ 5	> 5	≤ 5
RFR model 2	**Original**	0.65	0.06	1.02	0.24	4	3	50.3	94.4
	Tuned	0.67	0.05	1.05	0.24	4	2	48.5	94.6
GBR model 2	**Original**	0.98	0.11	1.31	0.33	5	2	27.0	89.6
	Tuned	0.62	0.06	1.00	0.24	4	3	52.3	94.5
SVR model 3	**Original**	1.14	0.13	1.46	0.37	4	2	21.1	86.7
	Tuned	1.02	0.14	1.36	0.38	4	2	27.2	85.9

5.2 Error analysis

5.2.1 Cumulative errors

In order to gain better insight into the performance of the tuned models, their cumulative error distributions were analysed (Figure 5.2). These plots show the fraction of buildings with an error less than or equal to a certain number of floors. If all buildings are considered (Figure 5.2a), the number of floors is predicted within 1 floor of the true value in approximately 99% of cases for all three models. RF and GB have an almost identical cumulative error distribution. These models perform better than SVR in terms of the fraction of predictions with an error of less than 1 floor.



(a) All buildings

(b) Above 5 floors

Figure 5.2: Cumulative errors of tuned models

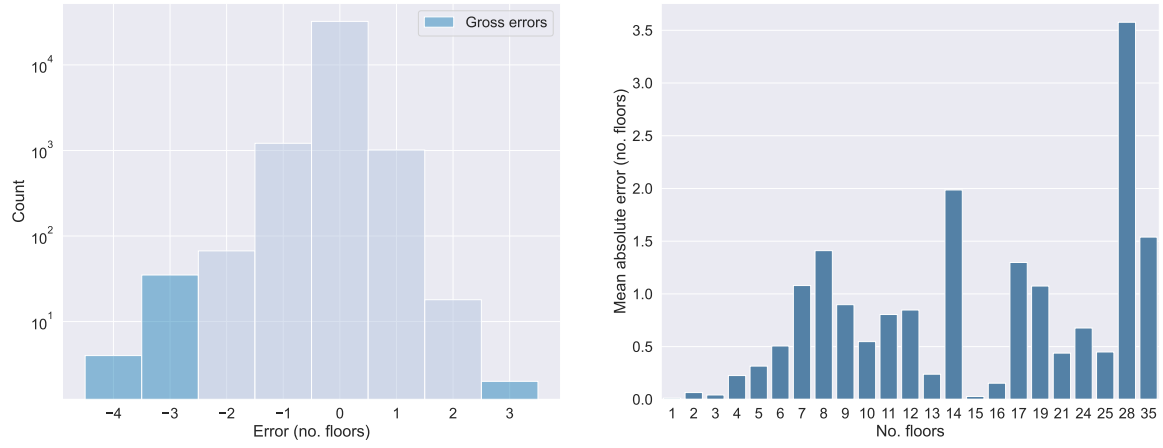
If buildings above 5 floors are considered separately (Figure 5.2b), the distribution shows somewhat greater differences between the models. RF and GB still have a very similar distribution, with approximately 80% of buildings predicted within 1 floor of the target for both models. Furthermore, around 90% of buildings have a prediction error of less than 1.5 floor, which corresponds to an error of 1 floor after rounding. SVR performs noticeably worse than the other two algorithms. The cumulative distribution does not approach 100% of buildings as quickly and the fraction of predictions with a certain absolute error is always lower. Nonetheless, more than 80% of buildings are predicted with an error of 1.5 floors or less, showing that the model still performs reasonably well above 5 floors.

The cumulative error distributions illustrate that there is little difference between the models obtained using RF and GB. This supports the results of the performance evaluation presented in Section 5.1.2. Since the tuned GB model performed slightly better above 5 floors, it was selected as the final model.

5.2.2 Gross errors

The cumulative error analysis demonstrated that, for the majority of buildings, the prediction error of the tuned GB model is relatively small. However, some buildings have a more substantial error of up to 4 floors. Errors above 2 floors were considered to be gross errors. This section analyses the gross errors of the tuned GB model in further detail. Based on the histogram provided in Figure 5.3a, it appears that most gross errors are caused by model underestimation. In contrast, prediction errors of 1 floor are balanced between over- and underestimation.

In order to gain a better understanding of where larger errors occur, the MAE was plotted for each number of floors (Figure 5.3b). This plot shows that the MAE is largest for buildings above 5 floors. The highest MAE is obtained for the buildings with 28 floors. In contrast, the model errors of lower storey buildings (below 6 floors) are almost negligible. These results make sense, as there were very few instances corresponding to high storey buildings in the training dataset. Too few training instances limit the ability of the model to generalise to unseen cases.



(a) Histogram of errors

(b) Mean absolute error per number of floors

Figure 5.3: Error distribution of best predictive model

Interestingly, there is an increase in the MAE between 6 to 8 floors, despite the fact that the training set contained many more examples of these floor counts than other higher storey buildings. The results of the semi-automatic data cleaning process (Figure 4.5) show that there is a substantial overlap in the height of buildings between 6–9 floors. Furthermore, the height range of buildings with 7–9 floors is much larger in comparison to other floor counts. This may have made it more difficult for the model to distinguish between these buildings, leading to larger errors.

Further analysis reveals the main cause of the gross errors. The majority of underestimated predictions correspond to incorrectly labelled buildings, which were not filtered out during the cleaning process. Since the model predicts (close to) the correct number of floors for these buildings, this led the error to appear higher than reality. Most incorrectly labelled buildings originated from the municipality of Rotterdam. A number of examples are provided in Figure 5.4. These examples provide an explanation for why the height range of buildings with 7–9 floors appeared broader than other floor counts. The assigned labels are (almost) twice the actual number of floors, while building height was measured correctly. This incorrect labelling may be because the floor count was determined by aggregating the number of floors of a footprint's constituent building units. If multiple units exist on the same floor, this would cause the number of floors to be incorrectly registered.

(a) Label: 7 floors; Actual: 4 floors; Predicted: 4 floors

(b) Label: 8 floors; Actual: 4 floors; Predicted: 5 floors

(c) Label: 9 floors; Actual: 5 floors; Predicted: 5 floors

Figure 5.4: Examples of incorrectly labelled buildings

The remaining gross errors mainly occurred for high storey apartment blocks (above 14 floors) and buildings with exceptionally large storey heights. For example, one of the royal palaces in Den Haag is predicted to have 7 floors, which is 3 floors higher than reality. As shown in Figure 5.5a, the storey height of this building is much larger than a normal four-storey building. Furthermore, the roof has a dome structure in the centre, which increases the measured building height. Similarly, the number of floors of high storey apartment blocks is also overestimated when building height is higher than would be expected for a specific number of floors. Further analysis shows that this may be because buildings situated beside water appear to have a lower ground level than reality (Figure 5.5b). This would increase the measured height of a building, particularly in combination with elevator shafts and other roof structures. Since the model was trained on too few instances of high storey buildings, this further decreased the model's ability to make correct predictions for these buildings.

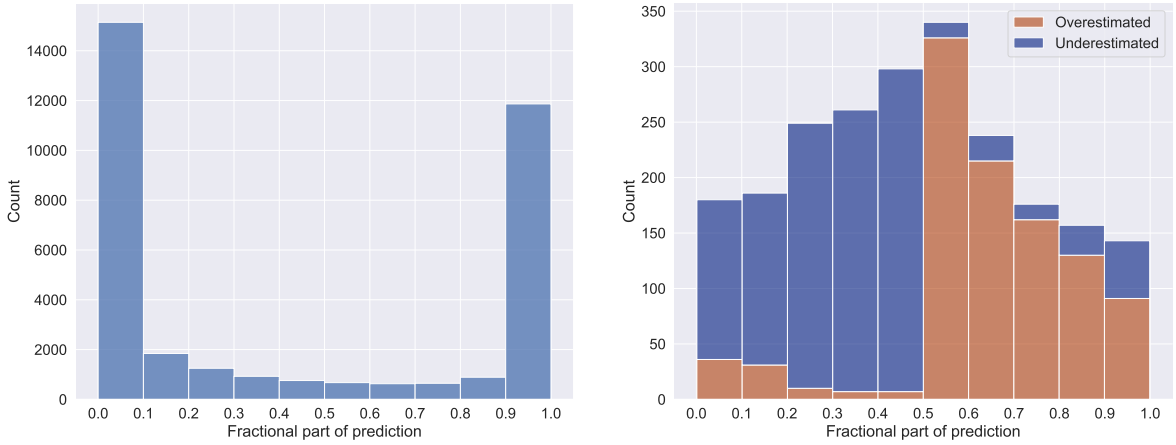
(a) Label: 4 floors; Actual: 4 floors; Predicted: 7 floors

(b) Label: 14 floors; Actual: 15 floors; Predicted: 17 floors

Figure 5.5: Examples of buildings with incorrect predictions

5.3 Impact of rounding

In order to determine to what extent the results were affected by rounding, the fractional part of the predictions was analysed. This corresponds to the predicted value minus the integer obtained by rounding the prediction down. The distribution of the fractional part of all predictions is shown in Figure 5.6a. It is interesting to observe that most predictions have a fractional part of either below 0.1 or above 0.9. This suggests that the rounding strategy did not have a large impact on the results, as the majority of predictions were already very close to an integer value.



(a) All predictions

(b) Rounded predictions with absolute error of 1 floor

Figure 5.6: Analysis of fractional part of predictions

It is also interesting to determine how often the fractional part caused the model to over- or underestimate the target value after rounding. This was achieved by analysing the fractional part of predictions that had an absolute error of 1 floor after rounding (Figure 5.6b). The results show that increasingly more predictions were over- or underestimated the closer the fractional part got to the halfway point between two integers, which makes sense as this is the most ambiguous case. The distribution is quite balanced, meaning that rounding caused a similar number of over- and underestimations.

Furthermore, an overestimation of 1 floor after rounding was mainly caused by fractional parts of above 0.5. Relatively few cases were overestimated by 1 floor because the fractional part was less than 0.5. This occurs when the predicted value is 1 to 1.5 floors larger than the target (e.g. a prediction of 3.1 for a target value of 2). Conversely, an underestimation of 1 floor after rounding was mainly caused by fractional parts below 0.5. Relatively few cases were underestimated by 1 floor because the predicted value was 1 to 1.5 floors smaller than the target (e.g. a prediction of 1.8 for a target value of 3).

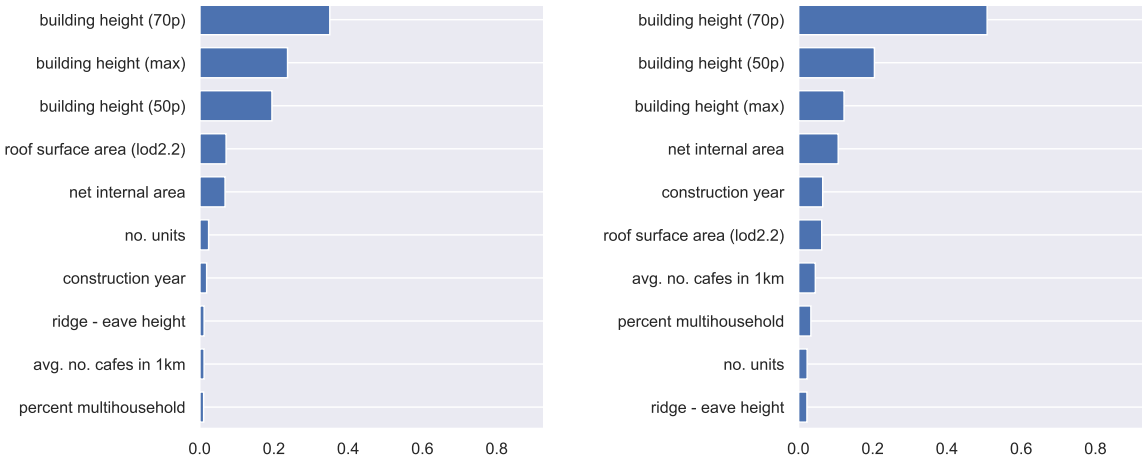
5.4 Feature contributions

The contribution of the individual features to the final model was assessed in terms of the impurity-based and permutation importance (Figure 5.7). As discussed in Section 3.4, permutation importance is a more reliable measure of feature contributions. This is because the importance scores are unbiased, unlike the impurity-based scores.

The results of both importance measures show that the different building height references were clearly the most important features. Building height at the 70th percentile was ranked highest, followed by the maximum and 50th percentile. Permutation importance considered 50th percentile height more important than maximum height, whereas this was the opposite for the impurity-based importance. Furthermore, the importance of the three height references was quite similar for the impurity-based

importance. In contrast, permutation importance considered 70th percentile height to be more than twice as important than the other height references.

Aside from building height, the impurity-based importance considered NIA and roof surface area to be slightly more important than the other features. These features were also considered to have a relatively high contribution to the model by the permutation importance. In addition, the permutation importance considered construction year to have an almost equal importance to roof surface area. This could be because it is related to storey height (as discussed in Table 3.1). When all features were considered, building height, roof surface area and NIA also ranked highest (Figure C.4). The importance of roof surface area was approximately the same for both LODs. However, the LOD1.2 versions of building volume and wall surface area ranked higher than their LOD2.2 equivalents.



(a) Impurity-based importance

(b) Permutation importance

Figure 5.7: Feature importance of best predictive model (tuned GB model 2)

The remaining features were assigned little importance based on impurity. The decrease in importance was less distinct based on permutation. In addition, the remaining features were ranked slightly differently by both importance measures. However, both metrics considered the difference between ridge and eave height to be one of the least important features. The census features also had a lower importance in both cases, which makes sense as these were only defined at a neighbourhood level.

5.5 Impact of data availability

The impact of data availability was assessed by training the best algorithm (GB after tuning) on different categories of features. This provides an indication of how well the model can be expected to perform when certain datasets are unavailable. This is interesting to consider because some datasets, such as a nationwide LOD2 model, are unavailable in other countries. Furthermore, this provides another perspective on how different features contribute to the model.

The results are shown in Table 5.4. For reference, the last row shows the performance of a model that predicts the mean number of floors in the dataset for all buildings.

The first model was based on all features in order to establish a baseline scenario in which all datasets are available. The models with the most comparable performance to this baseline were those based on the LOD1.2 and LOD2.2 features. The model based on LOD2.2 performed slightly better than LOD1.2. However, overall, there was very little difference in performance, showing that a higher level of detail is not required. A similar quality of predictions can be obtained based on LOD1.2.

The model based on cadastral features had the next best performance. For buildings below 5 floors, the MAE was almost double that obtained using the 3D geometric features. In addition, the MAE above 5 floors was almost half a floor higher. On the other hand, the accuracy was only reduced by around 8%

and was more than 80% for buildings with less than 5 floors. This shows that the 3D geometric features are more useful for reducing the prediction error, but a reasonably good level of accuracy can still be achieved with just cadastral features.

The worst performing models were those based on the census and 2D geometric features. These models provided an accuracy of around 60–65% below 5 floors, but only 5% or less for higher storey buildings. For buildings below 5 floors, the MAE of these models was almost the same, suggesting that these feature categories contribute a similar amount to model performance. However, the accuracy of the model based on census features was lower. This model also made larger prediction errors for buildings above 5 floors, showing that the 2D geometric features are slightly more useful. This makes sense as the 2D geometric features were extracted per footprint, whereas the census features were only available at a neighbourhood level.

An accuracy of around 60% is still quite high, which is particularly surprising for the model based on neighbourhood census data. This can be explained because a high level of accuracy (55.5%) can still be achieved if the mean number of floors is predicted for all buildings. However, the subset based on census features still achieves a 5% higher accuracy than the model based on the mean. This shows that the additional context that these features provide about the building's surrounding environment helps to improve model performance.

Table 5.4: Impact of different feature subsets on model performance

	Features					**Model performance**			
	Cadastral	**Geometric**			**Census**	**MAE**		**Accuracy (%)**	
		2D	**LOD1.2**	**LOD2.2**		**>5**	**≤ 5**	**>5**	**≤ 5**
1	×	×	×	×	×	0.64	0.05	51.7	94.8
2	×					1.35	0.19	25.3	82.5
3		×				2.23	0.39	5.8	65.2
4			×			0.89	0.10	32.5	90.1
5				×		0.87	0.10	34.8	90.5
6					×	2.55	0.41	3.6	61.7
Mean						3.95	0.52	0.0	55.5

Based on this analysis, the 3D geometric features appear to contribute most to the model. However, the model based on all features still provided notably better results. This suggests that combining the 3D geometric features with features derived from other datasets provides the best results. The LOD1.2 and cadastral features appear to be a good combination, since these perform well and are easy to obtain in the Netherlands. However, the availability of cadastral attributes is often lower in other countries, meaning a combination of LOD1.2 with 2D geometric features would be more widely applicable.

5.6 Comparison to purely geometric approach

In order to determine whether machine learning provides better results, the performance of the best predictive model was compared to the geometric approach. The results are outlined in Table 5.5. These results show that the best model performs better according to all four error metrics, apart from the maximum error of buildings below 5 floors. For buildings below 5 floors, the best model has a much lower MAE and is approximately 25% more accurate than the geometric approach.

Nonetheless, the geometric approach still provides relatively good predictions below 5 floors, with an accuracy of 70% and a maximum error of 2 floors. This means that, depending on the application, using the geometric approach could be sufficient. In addition, for buildings above 5 floors, the performance difference is less notable. However, machine learning has the potential to perform better if sufficient training data was available for higher storey buildings.

Table 5.5: Comparison of best predictive model to geometric approach above and below 5 floors

	MAE		RMSE		Max. error		Accuracy (%)	
	> 5	≤ 5	> 5	≤ 5	> 5	≤ 5	> 5	≤ 5
GB model 2 (tuned)	0.62	0.06	1.00	0.24	4	3	52.3	94.5
Geometric approach	0.70	0.31	1.09	0.31	5	2	47.5	69.9

It is interesting to note that, similar to machine learning, the geometric approach performs worse for higher storey buildings (Figure 5.8). This suggests that the number of floors of these buildings is inherently more difficult to predict. This could be due to building characteristics, such as the presence of elevator shafts or double storey ground floor lobbies. The input data could also play a role. The NIA may exclude larger sections of the actual internal area of these buildings due to, for instance, corridors and stairwells. As a result, machine learning may require more training instances of high storey buildings to reach a similar level of performance to lower storey buildings.

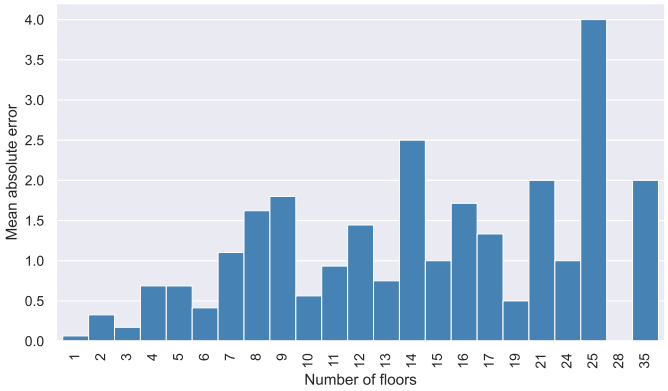
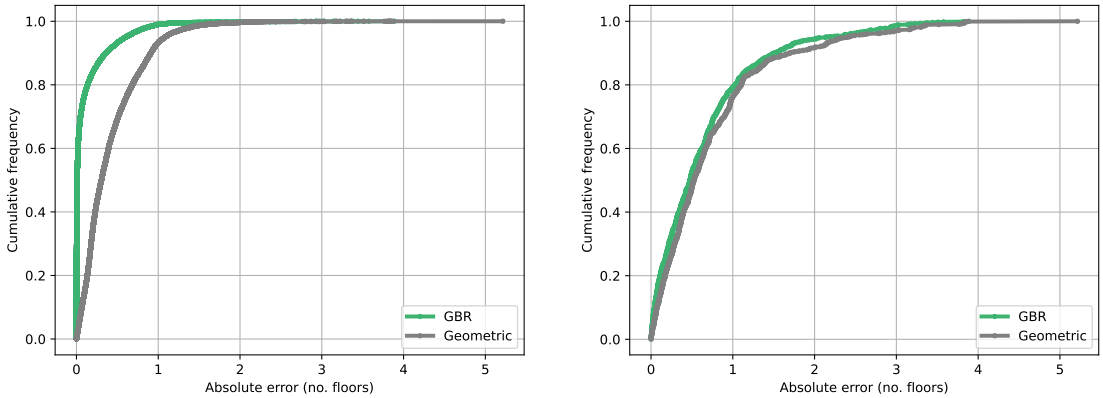


Figure 5.8: Mean absolute error per number of floors for the geometric approach

In addition to the performance evaluation, the cumulative error distributions of the two approaches were compared (Figure 5.9). Although machine learning provides more accurate predictions overall, the geometric approach estimates the number of floors within 1 floor of the target value for more than 90% of buildings (Figure 5.9a). This is almost the same as the best predictive model.



(a) All buildings

(b) Above 5 floors

Figure 5.9: Cumulative errors of best predictive model compared to geometric approach

If only buildings above 5 floors are considered (Figure 5.9b), the cumulative error distributions are almost identical. This shows that the best predictive model does not provide a substantial improvement on the current estimate. Furthermore, the geometric approach is sufficient for applications where an average error of one floor is acceptable.

Table 5.6: Comparison of results for case study buildings categorised by roof type

	Roof type		
Municipality	**Slanted**	**Single horizontal**	**Multiple horizontal**
Amsterdam			
	Actual no. floors: 4 Machine learning: 5 Geometric approach: 5	Actual no. floors: 5–6 Machine learning: 5 Geometric approach: 7	Actual no. floors: 15 Machine learning: 17 Geometric approach: 18
Rotterdam			
	Actual no. floors: 3 Machine learning: 3 Geometric approach: 3	Actual no. floors: 5 Machine learning: 5 Geometric approach: 6	Actual no. floors: 35 Machine learning: 33 Geometric approach: 37
Den Haag			
	Actual no. floors: 2 Machine learning: 2 Geometric approach: 3	Actual no. floors: 3 Machine learning: 3 Geometric approach: 4	Actual no. floors: 8 Machine learning: 7 Geometric approach: 9
Rijssen-Holten			
	Actual no. floors: 1 Machine learning: 1 Geometric approach: 2	Actual no. floors: 2 Machine learning: 2 Geometric approach: 2	Actual no. floors: 3 Machine learning: 3 Geometric approach: 3

Finally, to gain a more concrete understanding of where machine learning performs better than the geometric approach, a number of case study buildings were analysed. This analysis was also conducted in order to ensure that the performance differences were not always caused by incorrectly labelled buildings. The results are shown in Table 5.6.

The results show that machine learning provided a better estimate of the number of floors for buildings with larger than average storey heights. For example, the 5 storey building in Rotterdam and 3 storey building in Den Haag have slightly higher storeys height than average. These examples were predicted correctly by machine learning, whereas they were overestimated by the geometric approach.

In addition, machine learning provided a more accurate estimate for the 5 storey building in Amsterdam with a cafe on the ground floor. The neighbouring buildings are purely residential and have 6 floors, whereas the building with the cafe has only 5 floors due to the double height ground floor. The geometric approach could not distinguish this case and provided an overestimate of 2 floors.

Machine learning was also able to distinguish buildings with and without storeys beneath slanted roofs. For instance, the number of floors of the 1 storey building with a slanted roof in Rijssen-Holten was predicted correctly. This example was overestimated by the geometric approach because the slanted roof increased the measured height of the building. The slanted roof of the 2 storey building in Den Haag also caused the geometric approach to overestimate the number of floors.

Both approaches were unable to correctly determine the floor count of the 4 storey building in the centre of Amsterdam. The number of floors was overestimated by 1 floor in both cases. This is most likely due to the differences in storey height throughout the building. Furthermore, both approaches performed worst for the examples of high storey apartment blocks. Machine learning generally underestimated the number of floors of these buildings, whereas the geometric approach provided an overestimate.

5.7 Model application

The best predictive model was applied to (mixed-)residential buildings in three other municipalities. This allowed the applicability of the model to other areas of the Netherlands to be analysed. The municipalities of Delft, Albrandswaard and Renkum were chosen as case studies (Figure 5.10). Delft covers a more densely populated area between Rotterdam and Den Haag, while the municipalities of Albrandswaard and Renkum are less urbanised. Albrandswaard is located immediately to the south of Rotterdam, while Renkum is situated in the east of the Netherlands, close to the German border.

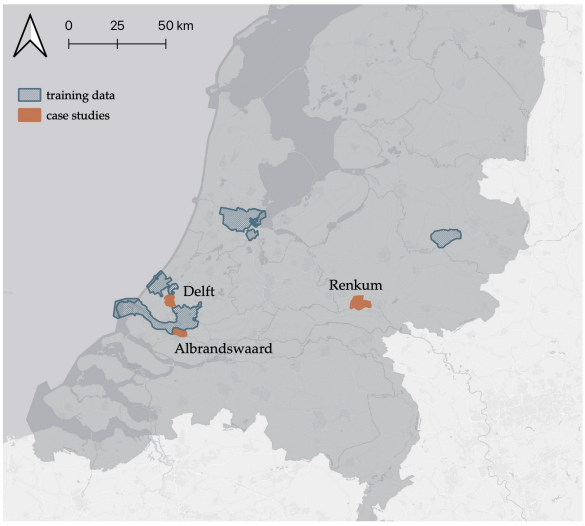


Figure 5.10: Map of case study and training data municipalities

Since the ground truth number of floors was not available for the case study municipalities, the performance of the model could not be determined. However, the predictions were compared to the geometric approach. A map comparison shows that the results are very similar (Figure 5.11 and Figure C.5). The geometric approach was found to be reasonably accurate in Section 5.6, particularly for buildings with less than 5 floors. Therefore, the similarity in the results suggests that machine learning still provides accurate predictions for other municipalities in the Netherlands.

Figure 5.11: Map of absolute difference between machine learning predictions and geometric approach for buildings in Delft

In addition to the map comparison, a number of individual buildings were analysed in further detail. This analysis focused on predictions that differed from the results of the geometric approach. Firstly, high storey apartment blocks in Delft were analysed. In some cases, machine learning provided an estimate that was closer to the true value (Figure 5.12a). However, similar to the example discussed in Section 5.2.2, overestimation seemed to occur for apartment blocks located above water. Furthermore, the number of floors of was generally still predicted incorrectly (Figure 5.12b). The worst prediction was obtained for a recently constructed apartment block, for which height was unavailable (Figure 5.12c). In this case, the geometric approach provided a much closer estimate based on NIA.

There were a number of cases where machine learning provided a better estimate of the number of floors. Similar to Section 5.6, this mainly occurred for buildings with storey heights that were larger than average, such as an old farmhouse in Rhoon (Figure 5.13a). In some cases, more accurate predictions were also obtained for buildings without storeys beneath slanted roofs (Figure 5.13b). However, in other cases, the number of floors was still overestimated (Figure 5.13c).

Overall, the analysis showed that the model was still applicable to other areas of the Netherlands. Furthermore, machine learning provided better predictions than the geometric approach for some more challenging cases. Lánský [2020] showed that a RF algorithm could be used to predict the height of 125

million building footprints in the USA in under 6 minutes by running processes in parallel on several CPUs. Therefore, the number of floors of building footprints in the Netherlands (approximately 10 million) could also be easily predicted within a reasonable amount of time. However, the additional time required to prepare the data seems disproportional to the increase in accuracy achieved in comparison to a purely geometric approach.

(a) Apartment above water
Actual no. floors: 16
Machine learning: 18
Geometric approach: 21

(b) Gallery flat
Actual no. floors: 18
Machine learning: 17
Geometric approach: 19

(c) Recently constructed apartment
Actual no. floors: 22
Machine learning: 4
Geometric approach: 15

Figure 5.12: Case studies of high storey apartments in Delft

(a) Rhoon, Albrandswaard
Actual no. floors: 3
Machine learning: 3
Geometric approach: 4

(b) Rhoon, Albrandswaard
Actual no. floors: 3
Machine learning: 3
Geometric approach: 4

(c) Heveadorp, Renkum
Actual no. floors: 2
Machine learning: 3
Geometric approach: 3

Figure 5.13: Case study buildings in Albrandswaard and Renkum

6 Discussion and conclusion

This chapter presents the conclusions of the analysis. The research questions are reviewed, in order to determine to what extent the objectives of this thesis were fulfilled. After this, the main contributions are discussed, as well as the limitations of the approach. Based on this, recommendations are provided and new research questions that emerged during the analysis are discussed.

6.1 Research overview

The aim of this thesis was to develop an alternative method to automatically infer the number of floors, with a focus on (mixed-)residential buildings in the Netherlands. This aim was developed due to the limitations of using a purely geometric approach. Since similar studies had demonstrated the potential of machine learning to infer building properties, the alternative method was also developed based on machine learning. One main research question was defined, in addition to three sub-questions (Section 1.1). These research questions are reviewed below.

Main research question: To what extent can machine learning provide a better estimate of the number of floors than a purely geometric approach?

The analysis has shown that machine learning can partially provide a better estimate of the number of floors, given the data available during this thesis. The best model mainly improved the predictions obtained for buildings with 5 floors or less. In particular, a better estimate was obtained for low storey buildings with larger than average storey heights. Furthermore, in some cases, the model was better able to distinguish buildings with and without storeys beneath slanted roofs. However, the results obtained for higher storey buildings (above 5 floors) were not substantially better than those obtained by the geometric approach. This is mainly because the training dataset was not representative of these buildings. Around 90% of the available training data consisted of buildings below 5 floors. As a result, it was more difficult for machine learning to infer patterns for higher storey buildings. This shows that better predictions can only be obtained if sufficient training instances are available.

a. Which features are related to the number of floors? Is there any overlap between these features and which subset yields the best results?

In total, 25 features were derived from cadastral attributes, building geometry and neighbourhood census data. The geometric features were extracted from both 2D building footprints and 3D building models at different levels of detail (LOD1.2 and 2.2). In addition, four percentiles were used to represent different geometric references for building height.

The analysis demonstrated that building height is most related to the number of floors, specifically 70th percentile height. This feature showed the strongest relationship to the number of floors in terms of the Pearson correlation coefficient and Mutual Information score. It was also ranked highest according to the permutation and impurity-based importance measures. Maximum and 50th percentile building height were also found to be strongly related to the number of floors, but less than the 70th percentile. Overall, the results showed that building height is considerably more relevant to the prediction problem than any of the other features. Nonetheless, a number of other 3D geometric features were found to be quite closely related to the number of floors, specifically roof area and volume. Cadastral features were also found to be relevant; mainly NIA and, to a lesser extent, construction year.

Hierarchical clustering based on Ward's linkage was used to determine which features were correlated with each other. This analysis found that there is substantial overlap between many of the 3D geometric features, most notably roof surface area and volume in both LOD1.2 and 2.2 and wall surface area in LOD1.2. These features also overlapped with the NIA and number of units. Unsurprisingly, the building height references were also found to be closely related to each other, but formed a separate cluster from the other 3D geometric features. The census features were also quite closely related. However, it should be noted that the results of this analysis were dependent on the linkage function used. Different linkage functions may have highlighted other overlaps between features, but this was not investigated.

Three feature selection methods were used to reduce the number of input features to a smaller subset. However, the results obtained for each subset did not differ substantially from each other. This made it difficult to determine which subset yields the best results for this prediction problem. All feature subsets included features related to building height and 3D geometry. Most subsets also included cadastral features and census or 2D geometric features. These subsets performed better than those based on single categories of features (e.g. only cadastral or 3D geometric). This suggests that using a combination of different types of features provides the best results, as this allows the data to provide the broadest description of each building.

b. Which machine learning algorithm provides the best results? How are the results affected by feature subsets that reflect different levels of data availability?

Three different algorithms were used in this thesis: Random Forest (RF), Gradient Boosting (GB) and Linear Support Vector Regression (SVR). The tree-based algorithms (RF and GB) provided the best results. Linear SVR performed substantially worse than the other two algorithms, even after tuning the hyperparameters. This suggests that a non-linear model is more suitable for this prediction problem. After tuning, GB provided slightly better results than RF and was selected as the final model.

The effect of data availability was investigated by comparing the performance of models based on different categories of features. In comparison to a model trained on all features, the performance decreased least for models based on 3D geometry. Interestingly, the results were almost the same for models based on LOD1.2 and LOD2.2, showing that a higher level of detail is not required. The level of performance decreased further if only cadastral features were used. However, relatively good performance was still achieved for buildings with 5 floors or less. The worst predictions were obtained if only 2D geometric or census features were used, showing these had the least relevance to the number of floors.

c. What level of performance can be achieved compared to a purely geometric approach? What types of gross errors are present?

Model performance was assessed using four different error metrics: MAE, RMSE, maximum error and accuracy. Accuracy is a classification metric, but was still relevant in this case because the predictions and labels were both integers (see Section 3.3.2). RMSE was mainly included to provide better insight into the effect of different model parameters, since it gives more weight to large errors. The results show that, for buildings with 5 floors or less, the best predictive model achieved an accuracy of 94.5% and a MAE of 0.06. Above 5 floors, model performance was substantially lower, with an accuracy of 52.3% and MAE of 0.62. The maximum error obtained above 5 floors was also one floor higher. However, an analysis of the gross errors found that most large errors were caused by incorrectly labelled buildings. Despite the incorrect labels, the number of floors of these buildings had been predicted correctly. The remaining gross errors mainly corresponded to incorrect predictions for high storey buildings.

In comparison to the geometric approach, the results obtained for buildings with 5 floors or less were significantly better. The accuracy of the geometric approach was 69.9% for these buildings, which is around 25% lower than the best predictive model. However, above 5 floors, there was a less notable difference in performance, with both approaches achieving an accuracy of around 50%. In addition, a comparison of the cumulative errors showed that the distributions were almost identical for buildings above 5 floors. This showed that the best predictive model did not provide a substantial improvement on the current estimate. Furthermore, the geometric approach estimated the number of floors within one floor of the target value for more than 90% of buildings. Machine learning mainly improved the fraction of buildings that were predicted with an error of less than 1 floor.

d. Since floor count is generally an integer value, is this a regression or classification problem? If considered regression, how does rounding the predictions affect the results?

In theory, the number of floors could be predicted as discrete classes by a classification algorithm or obtained after rounding the predictions of a regression algorithm. However, early on in the analysis (Section 2.3.1), it was determined that it is more appropriate to consider this a regression problem. This is because classification would require the training data to include examples of all possible floor counts that exist in reality. Without this information, the model would be unable to predict the number of floors of the missing classes. In practice, it would be difficult to find this data. For instance, the training data available during this thesis did not include examples of buildings with 26 to 44 floors.

In contrast, the predictions of a regression model are not limited to the training data examples, allowing predictions to be made for unseen cases. This also allows a regression model to remain applicable to new buildings constructed with increasingly higher floor counts. Furthermore, a regression model would take into account the ordinal nature of the number of floors, whereas in a classification model each floor count would represent an isolated class. A further advantage is that the number of floors would be predicted as a decimal value by regression, allowing buildings with half floors to be taken into account. This is required for some applications, such as energy demand estimation.

Since training data was only available for buildings with whole numbers of floors, the predictions were rounded to the nearest integer. The impact of this rounding strategy was investigated by analysing the distribution of the fractional part of the predictions. This analysis showed that the fractional part of the majority of predictions was either below 0.1 or above 0.9, meaning that most results were already close to the nearest integer. As a result, rounding did not have a large impact on the results. Furthermore, the analysis investigated the distribution of the fractional part of predictions with an absolute error of one floor after rounding. The distribution was quite balanced, meaning that rounding caused a similar amount of over- and underpredictions.

6.2 Discussion

6.2.1 Contributions

This thesis builds upon the results of previous studies into the generation and enrichment of 3D building models using machine learning. An alternative approach to infer the number of floors was developed, which provided a partial improvement on the results of a purely geometric approach. Overall, there are three main areas where the analysis provided additional insight into this prediction problem:

- **Reliability of geometric approach**: Prior to this thesis, the reliability of using a purley geometric approach had only been assessed on a small scale through the inspection of a number of case study buildings in Google Street View. This thesis has investigated the geometric approach on a much larger scale, using different error metrics to quantify the level of performance. This has allowed a better understanding of its reliability to be gained. The results can be used as a baseline to compare the performance of future methods to.
- **Analysis of contributing factors**: A variety of features were assessed, covering different categories of data (cadastral, geometric and census). The analysis provided a better understanding of which features were most related to the number of floors and the impact of different feature combinations on the results. In addition, the comparison of the 3D geometric features provided valuable insight, as it showed that a higher level of detail did not improve the results. This is a useful outcome because it shows that reliable predictions could still be obtained in areas where only simple building models are available.
- **Classification vs. regression**: This thesis has provided clear reasoning for why inferring the number of floors should be considered a regression problem. Furthermore, the analysis has shown that rounding the predictions to the nearest integer does not necessarily lead to an artificial improvement of the results, since the majority of predictions were already close to a whole number of floors.

6.2.2 Limitations

Although this thesis has shown that machine learning can partially provide a better estimate of the number of floors, there are a number of limitations to the approach; outlined as follows:

- **Training data**: The methodology relied on data obtained from the BAG+ datasets of individual municipalities. In some cases, this data was not reliable and a large amount of additional processing time was required for cleaning. Furthermore, the data was heavily skewed towards lower storey buildings, which prevented reliable results from being obtained for buildings above 5 floors. In addition, no data was available for buildings with half floors, preventing the applicability of the model to these buildings from being investigated.
- **Data cleaning**: In order to ensure that the algorithms were trained on reliable data, much of the data preparation process focused on data cleaning. In some ways, this process was too excessive, causing a large amount of data to be removed. In addition, a number of assumptions were made in order to perform the (semi-)automatic steps. These assumptions may have led the model to become more biased towards certain features. For instance, the distribution of 70th percentile height was used to clean the training labels, which may have caused this feature to seem more related to the number of floors than the maximum or 50th percentile height.
- **Feature extraction and selection**: Some aspects of the feature extraction process could be improved. In particular, the methodology used to extract ridge and eave height. This was not reliable for all buildings, which may have introduced additional noise into the data and negatively impacted the results. Furthermore, the analysis did not consider whether combining different features improved the results. In terms of feature selection, three different methods were used, resulting in 9 models. This complicated the analysis and ultimately did not lead to much difference in the results. It also seems likely that a better feature subset exists. Furthermore, the feature selection process was based on trends in the training dataset, which was skewed towards low storey buildings. A distinction was not made between features that were more relevant for certain types of buildings (e.g. high and low storey).
- **Model performance and results**: Model performance was assessed in terms of absolute measures of performance. However, it would have also been interesting to consider the relative error distribution, since an error of 1 floor is more significant for a one storey building than a high storey apartment block. Although the best predictive model provided better predictions for buildings with 5 floors or less, there was no significant improvement for buildings above 5 floors. Since a large amount of time was invested in the data processing steps, the level of improvement achieved does not seem proportional to the complexity of the approach. In addition, the influence of different height percentiles on the geometric approach was not considered. This means that better results may have been obtained for a different height percentile, further reducing the improvement provided by machine learning.

6.3 Recommendations and future work

As part of this discussion, a number of recommendations are provided for similar research based on machine learning. Firstly, in terms of training data, the possibility of obtaining additional data should be investigated, with a particular focus on higher storey buildings (above 5 floors). OpenStreetMap (OSM) could be a potential source of this data, in addition to the BAG+ datasets of other municipalities. Investigating the availability of data from neighbouring countries could also be useful, since building characteristics are fairly similar in these countries, particularly in terms of high storey buildings. However, it seems likely that the data will remain skewed towards low storey buildings, as these are more numerous in reality. Therefore, methods to take into account data imbalance should be considered.

Secondly, in terms of data cleaning, the process should avoid becoming too excessive. In comparison to this thesis, a more incremental approach should be used. The data cleaning steps could be conducted alongside the modelling process, allowing the impact on model performance to be assessed after each

cleaning step. This would prevent too much data from being removed and allow the cleaning process to focus on the most relevant aspects of the data. In addition, the relationship of the features to the target variable should be assessed before and after cleaning, in order to determine to what extent the assumptions made affected the results.

Thirdly, it is useful to extract a large variety of features based on different types of data. In this thesis, models based on a combination of different types of features provided better results than models trained on a single category. Therefore, it is advisable to test as many features as possible. Features that have low individual relevance could still have a high relevance when combined with other features.

In addition to these recommendations, a number of new research questions have emerged based on the results of the analysis. Possible directions for future work are outlined as follows:

- **Wider model applicability**: Firstly, the applicability of the model to buildings with half floors could be assessed, as well as the level of accuracy achieved for non-residential buildings. These two cases are important to consider since they are relevant for certain applications, such as energy demand estimation. Furthermore, the applicability of the model to other countries could also be investigated, as lack of open data on the number of floors is not limited to the Netherlands. Finally, future work could focus on determining whether machine learning could be used to infer the number of storeys below ground as well. This could be partly based on NIA, since this includes areas related to underground storeys.
- **Automatic data correction**: Since data on the number of floors obtained from the BAG+ datasets was not always reliable, it would be useful to investigate to what extent machine learning could be used to flag incorrect records and provide suggestions for automatic corrections. The buildings removed during the semi-automatic cleaning process could be analysed in further detail, in order to determine to what extent the model provided correct predictions for these cases.
- **Analysis of input features**: It would be useful to investigate whether combining different features would help to improve the results. For instance, footprint area may be more useful in combination with NIA, as this provides an approximate measure of the number of floors. Additional features could also be derived from other datasets, such as street networks, to increase the amount of input data. In addition, the relevance of the features to different parts of the data could be investigated. For instance, high and low storey buildings could be considered separately, or buildings from different construction periods. This could be used to develop a range of models that perform well on different parts of the data. These models could then be combined into an ensemble that would provide predictions that were applicable to any building.
- **Improved geometric approach**: The results of the analysis suggest that the geometric approach already provides a relatively good estimate of the number of floors, particularly for low storey buildings. Rather than focusing on machine learning, future research could investigate the potential of developing an improved geometric approach. Since a large amount of data on the number of floors has been collected, this could be used to analyse patterns for different types of buildings and construction periods. An unsupervised learning algorithm could potentially be used to pinpoint specific clusters of buildings with similar properties. Based on this, the most appropriate building height reference and storey height could be selected for each case. An appropriate offset factor could also be determined to take into account elevator shafts or double-height lobbies. This would allow the geometric approach to be adjusted to form a hierarchy of decisions related to building type and other characteristics.

A Dataset information

This appendix provides additional information about the datasets used to extract the features used by the machine learning algorithms. An overview is provided in Table A.1, including links to the datasets and metadata. All datasets are openly available apart from the dataset on building type maintained by ESRI, which is only accessible for ArcGIS users.

Table A.1: Overview of datasets used to extract training features

Dataset name	Description	Version	Source
BAG	National cadastral dataset	04-2020	(1)
3D BAG	3D building models of BAG footprints	21.03.1	(2)
Kerncijfers wijken en buurten	Neighbourhood census dataset maintained by the CBS	2019	(3)
Woningtypering	Building type of BAG footprints maintained by ESRI	2019	(4)

(1) https://data.overheid.nl/en/dataset/0ff83e1e-3db5-4975-b2c1-fbae6dd0d8e0

(2) https://doi.org/10.4121/uuid:f1f9759d-024a-492a-b821-07014dd6131c / https://3dbag.nl/en/download

(3) https://data.overheid.nl/dataset/09f5479a-50f9-45ed-b727-91bf141d14f4

(4) Data overview: https://www.arcgis.com/home/item.html?id=fa01ef63321e482e9b2c55620e554ffc

(5) Metadata: https://www.arcgis.com/sharing/rest/content/items/fa01ef63321e482e9b2c55620e554ffc/info/metadata/metadata.xml?format=default&output=html

B Additional feature extraction details

B.1 Selection of radius for number of neighbours

The number of neighbours within a certain radius of each building was computed and use as input to the machine learning algorithms (see Section 4.4). In order to determine the best radius to use, the results obtained for different radii were compared. The number of neighbours was computed for radii of 25, 50, 75 and 100 meters. Then, a RF algorithm was trained four times on the 2D geometric features, each time with a different choice of radius. The model evaluation results are provided in Table B.1.

The results obtained for each radius are quite similar. For building above 5 floors, the highest accuracy and lowest MAE were obtained for a radius of 50m. For buildings with 5 floors or less, the highest accuracy was obtained for a radius of 100 meters, while the lowest MAE was obtained for radii of 75 and 100 meters. Overall, the lowest maximum error was obtained for a radius of 50 meters.

Table B.1: Model evaluation results for the test set based on the no. of neighbours within different radii

	Accuracy (%)		**MAE**		**Max. error**
Radius	**> 5**	**≤ 5**	**> 5**	**≤ 5**	**All**
25m	9.4	66.7	2.24	0.38	27
50m	10.4	67	2.17	0.38	26
75m	8.4	67.6	2.22	0.37	29
100m	8.3	67.9	2.2	0.37	27

Based on these results, it still remained difficult to choose the best radius. Therefore, the permutation importance of each feature was also considered (see Section 3.4 for definition). The results are shown in Table B.2. Since a radius of 100 meters had a slightly higher permutation importance and also performed better for buildings with 5 floors or less, it was selected as the best choice of radius.

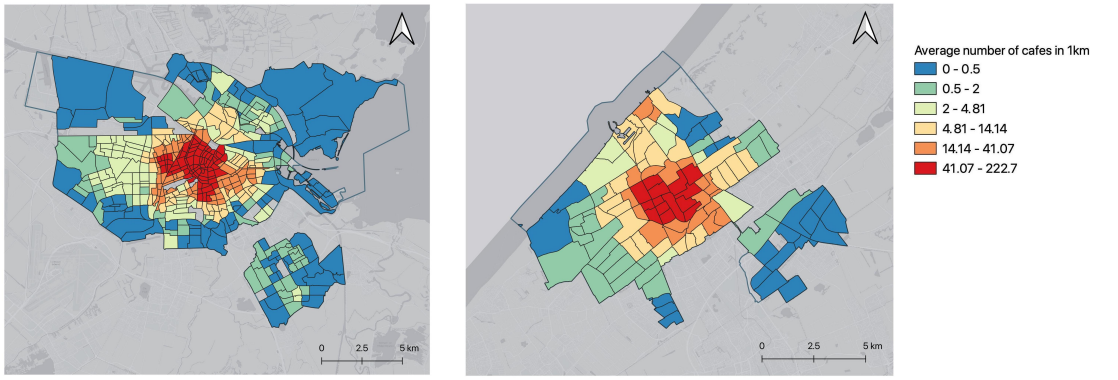
Table B.2: Permutation importance of the no. of neighbours within different radii

No. neighbours	**Importance**
25m	0.25
50m	0.27
75m	0.28
100m	0.29

B.2 Comparison of census features related to amenities

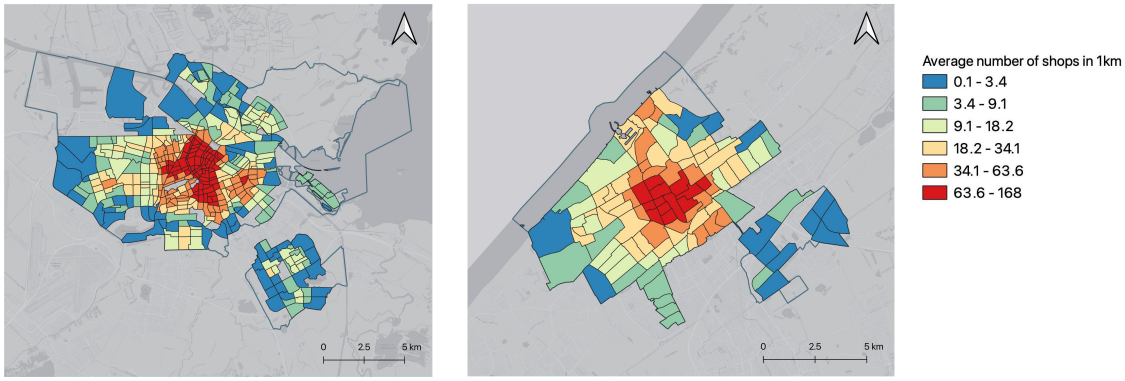
The CBS dataset on neighbourhood statistics contains a number of different attributes related to the average number of amenities within 1km. These attributes were considered useful as they could be used as input to the machine learning algorithms to represent the demand for amenities (see Section 4.4). In order to prevent the number of input features from becoming too large, only one of these attributes

was selected. The distribution of the average number of cafes, shops and supermarkets within 1km was compared (Figure B.1, Figure B.2, Figure B.3), allowing the most useful attribute to be selected. The average number of shops and cafes showed the strongest relationship to the area morphology, with a high number of these amenities located in the centre of each municipality and a lower number in rural and suburban areas. Since data on the average number of cafes was available for more statistical neighbourhoods, it was chosen as the most suitable attribute to represent demand for amenities.



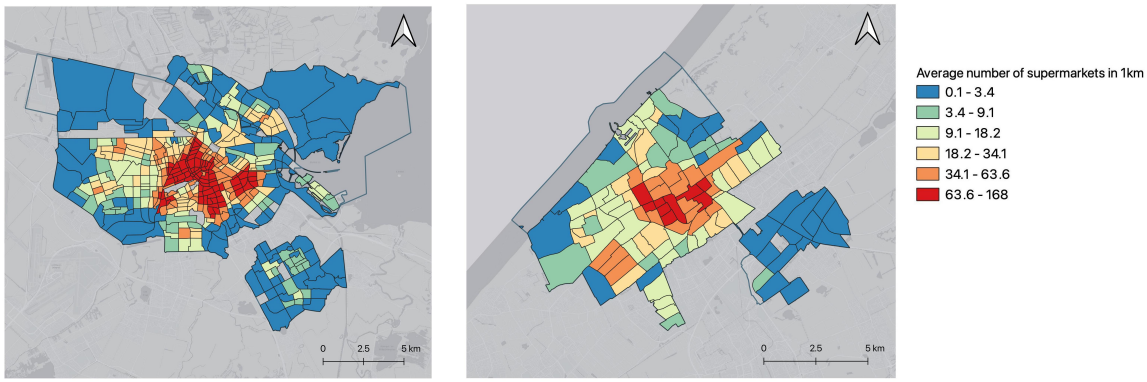
(a) Amsterdam (b) Den Haag

Figure B.1: Average number of cafes within 1km per statistical neighbourhood



(a) Amsterdam (b) Den Haag

Figure B.2: Average number of shops within 1km per statistical neighbourhood

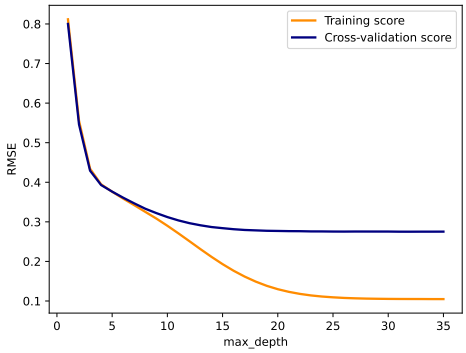


(a) Amsterdam (b) Den Haag

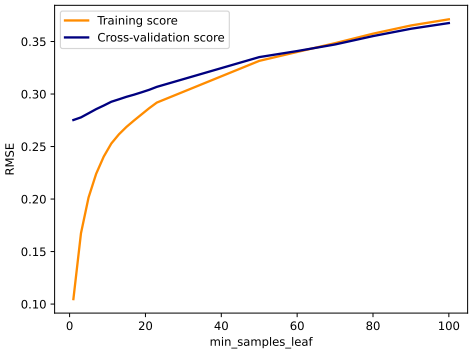
Figure B.3: Average number of supermarkets within 1km per statistical neighbourhood

C Additional machine learning results

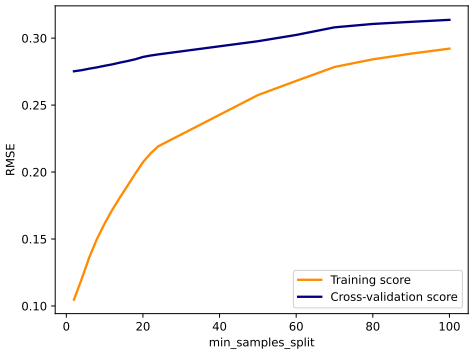
C.1 Hyperparameter tuning



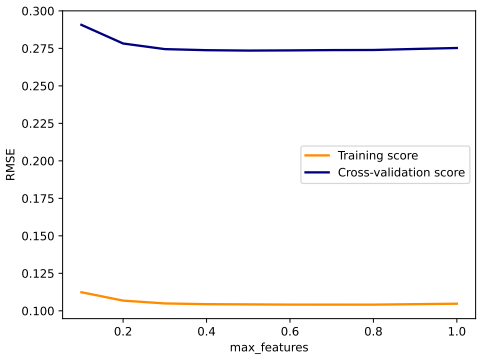
(a) Maximum tree depth



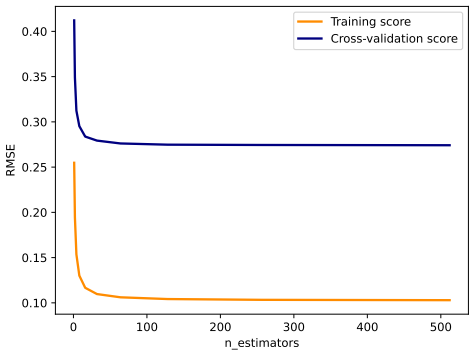
(b) Minimum no. samples of leaf nodes



(c) Minimum no. samples of internal nodes



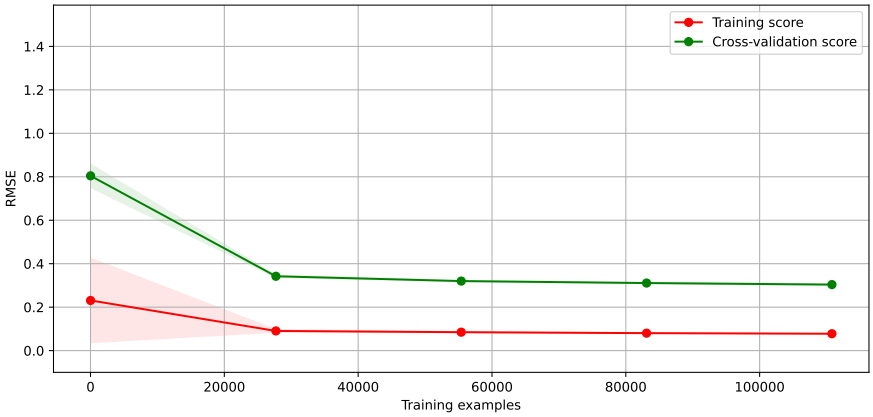
(d) Maximum no. features to consider



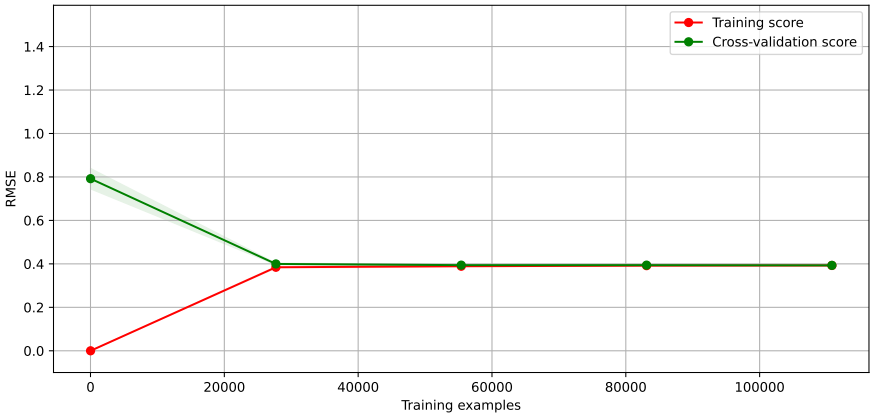
(e) No. trees in ensemble

Figure C.1: Validation curves for Random Forest hyperparameters

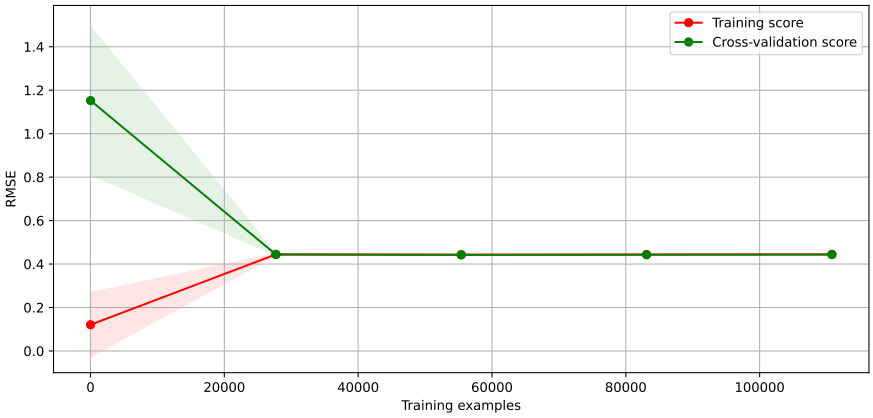
C.2 Model performance



(a) Random Forest



(b) Gradient Boosting



(c) Linear Support Vector Regression

Figure C.2: Learning curves before hyperparameter tuning for models based on all features

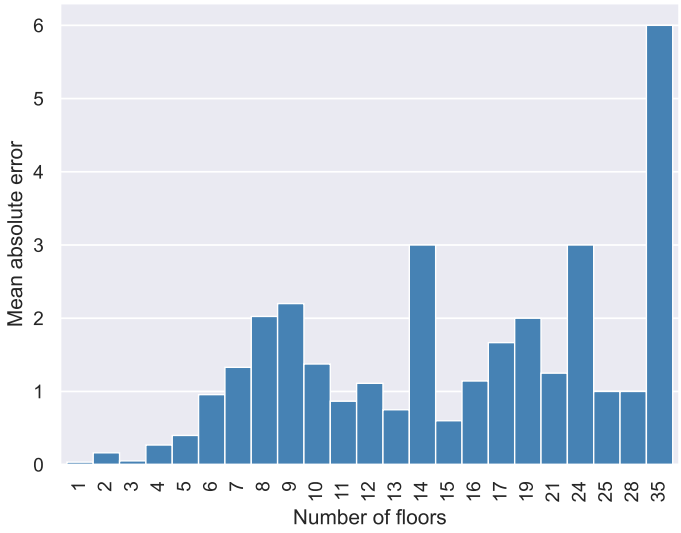
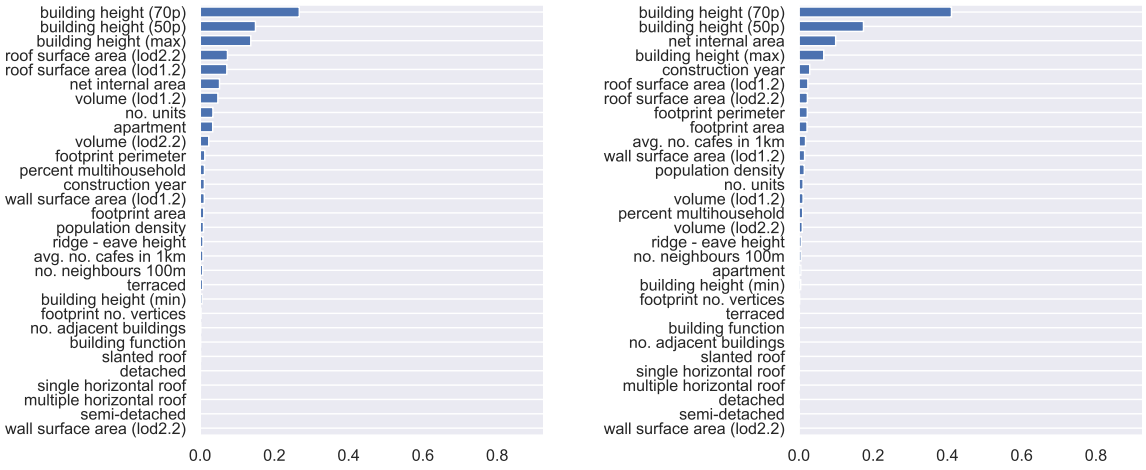


Figure C.3: Mean absolute error per number of floors for Support Vector Regression (all features)

C.3 Feature contributions



(a) Impurity-based importance

(b) Permutation importance

Figure C.4: Feature importance of best predictive model (GB after tuning) using all features

C.4 Model application

(a) Rhoon, Albrandswaard

(b) Doorwerth, Renkum

Figure C.5: Maps of absolute difference between machine learning predictions and geometric approach

D Reproducibility self-assessment

D.1 Marks for each of the criteria

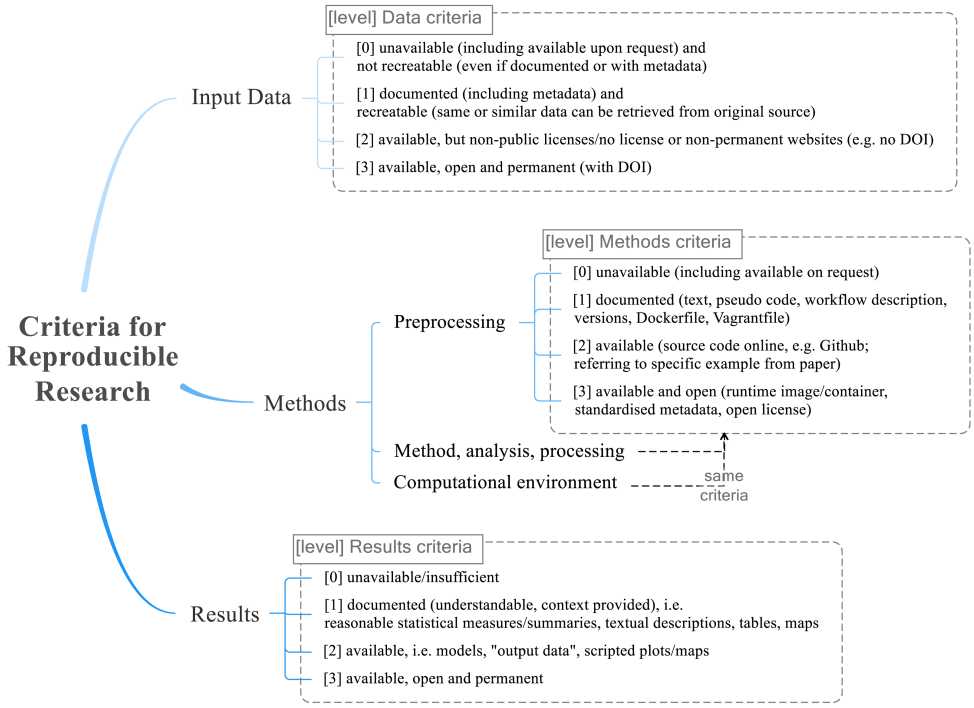


Figure D.1: Reproducibility criteria to be assessed.

Based on the reproducibility criteria, the following scores were assigned to each aspect of the research:

a. input data: 0-2

b. preprocessing: 1-2

c. methods: 1-2

d. computational environment: 1-2

e. results: 1-2

D.2 Self-reflection

The reproducibility of this thesis was assessed according to five criteria (Section D.1). Firstly, the availability of the input data was given a score of 0–2. Data on the number of floors was obtained from the BAG+ datasets of individual municipalities. The majority of this data is not openly available and could only be obtained upon request. Data on the number of floors for the municipality of Amsterdam

is openly available, but this dataset is frequently updated, meaning that only similar data could be retrieved. The majority of datasets used to extract features are openly available, but these datasets do not have a DOI, meaning the data could change or become unavailable in the future.

A large number of pre-processing steps were performed on the input data. These steps are documented in text and workflow descriptions in Chapter 3 and Chapter 4. In addition, the source code is openly available online via a Github repository, allowing many of the pre-processing steps to be directly reproduced. For instance, the automatic data cleaning process and feature selection approaches. However, some of the steps were performed manually (e.g. manual data cleaning). These steps are documented, but could not be directly reproduced. Therefore, this aspect was scored 1–2.

In terms of the method, analysis and processing, only open source tools and programming languages were used (e.g. Python, PostgreSQL). Proprietary software was avoided, apart from to load the data into the database (which was partially implemented using FME). The steps taken are thoroughly documented in Chapter 3 and Chapter 4. Furthermore, the source code of the implementation and most of the analysis is available via a Github repository. This repository also provides an overview of the software libraries (and versions) used, allowing the computational environment to be reproduced. Therefore, the second and third aspects were also given a score of 1–2.

Finally, the results were documented using statistical measures, textual descriptions, tables, maps and plots in Chapter 5. The majority of the results were obtained automatically using the code provided in the Github repository. This code includes the scripts that were used to generate the plots. However, the maps were generated manually using QGIS and would be less easy to recreate. Furthermore, the case study buildings were inspected manually and the ID or address of these buildings was not provided in the text, meaning the same analysis could not be easily performed. The model predictions were not made openly available because of the large file size. It is unclear whether this data could be made available upon request, since it is based on municipal records that are not publicly available. Based on this, the final aspect also received a score of 1–2.

Bibliography

Agugiaro, G. (2015). Energy planning tools and CityGML-based 3D virtual city models. Experiences from Trento (Italy). *Applied Geomatics*, 8.

Alahmadi, M., Atkinson, P., and Martin, D. (2013). Estimating the spatial distribution of the population of Riyadh, Saudi Arabia using remotely sensed built land cover and height data. *Computers, Environment and Urban Systems*, 41:167 – 176.

Alamdari, A. R. S. A. (2006). *Feature Extraction: Foundations and Applications*, chapter 14: Variable Selection using Correlation and Single Variable Classifier Methods: Applications, pages 343–358. Springer.

Anh, P., Thanh, N. T. N., Vu, C. T., Ha, N. V., and Hung, B. Q. (2018). Preliminary Result of 3D City Modelling For Hanoi, Vietnam. In *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 294–299.

Biljecki, F. and Dehbi, Y. (2019). RAISE THE ROOF: TOWARDS GENERATING LOD2 MODELS WITHOUT AERIAL SURVEYS USING MACHINE LEARNING. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W8:27–34.

Biljecki, F., Ledoux, H., and Stoter, J. (2014). Height references of CityGML LOD1 buildings and their influence on applications.

Biljecki, F., Ledoux, H., and Stoter, J. (2016a). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25 – 37.

Biljecki, F., Ledoux, H., and Stoter, J. (2017). Generating 3D city models without elevation data. *Computers, Environment and Urban Systems*, 64:1–18.

Biljecki, F., Ledoux, H., Stoter, J., and Vosselman, G. (2016b). The variants of an LOD of a 3D building model and their influence on spatial analyses. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:42 – 54.

Biljecki, F. and Sindram, M. (2017). Estimating building age with 3D GIS. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W5:17–24.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D city models: state of the art review. *ISPRS International Journal of Geo-Information*, 4:284–2889.

Boeters, R. (2013). Automatic enhancement of CityGML LoD2 models with interiors and its usability for net internal area determination. Master's thesis, Delft University of Technology.

Bousquet, O., von Luxburg, U., and Rätsch, G. (2004). *Advanced Lectures on Machine Learning*. Springer, Heidelberg, Germany.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (1997). Arcing the edge. Technical report, University of California.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Chai, T. and Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3):1247–1250.

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28. 40th-year commemorative issue.

Chen, C. and Breiman, L. (2004). Using Random Forest to Learn Imbalanced Data. *University of California, Berkeley*.

Duan, K., Keerthi, S. S., and Poo, A. N. (2003). Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59.

Duch, W. (2006). *Feature Extraction: Foundations and Applications*, chapter 3: Filter Methods, pages 89–117. Springer.

Dukai, B. (2018). 3D Registration of Buildings and Addresses (BAG) / 3D Basisregistratie Adressen en Gebouwen (BAG). 4TU.ResearchData. https://doi.org/10.4121/uuid:f1f9759d-024a-492a-b821-07014dd6131c.

Dukai, B., Ledoux, H., and Stoter, J. (2019). A Multi-Height LoD1 Model of all Buildings in the Netherlands. In *14th 3D GeoInfo Conference 2019*, volume IV-4 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 51–57. ISPRS.

Ellenkamp, Y. and Rietdijk, M. (2010). Kwaliteit van de basisregistraties adressenen gebouwen. Technical report, Ministerie van Volkshuisvesting, Ruimtelijke Ordening en Milieubeheer.

ESRI (2021). Woningtypering: Kaartlaag met de classificatie van woningtypes op basis van de BAG. https://www.arcgis.com/home/item.html?id=fa01ef63321e482e9b2c55620e554ffc (accessed 08.04.2021).

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232.

GDAL/OGR contributors (2021). GDAL/OGR Geospatial Data Abstraction software Library. Open Source Geospatial Foundation. https://gdal.org (accessed: 24.11.2021).

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow*. O'Reilly Media, Inc.

Gregorio, F. D. and Varrazzo, D. (2001–2021). Psycopg – PostgreSQL database adapter for Python. https://www.psycopg.org/docs/ (accessed: 24.11.2021).

Grömping, U. (2009). Variable Importance Assessment in Regression: Linear Regression versus Random Forest. *The American Statistician*, 63(4):308–319.

Gröger, G. and Plümer, L. (2012). CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33.

Guyon, I. and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157–1182.

Heeres, E. (2016). Exploring the 3D BAG: How to define it and to what extent can it automatically be created using open data. Master's thesis, Delft University of Technology.

Henn, A., Römer, C., Gröger, G., and Plümer, L. (2012). Automatic classification of building types in 3D city models. *Geoinformatica*, 16.

Hill, R. and Adkins, L. (2007). *A Companion to Theoretical Econometrics*, chapter 12: Collinearity, pages 256–278. Blackwell Publishing Ltd.

Hintze, J. L. and Nelson, R. D. (1998). Violin Plots: A Box Plot-Density Trace Synergism. *The American Statistician*, 52(2):181–184.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). *An Introduction to Statistical Learning*, chapter 3: Linear Regression, pages 59–128. Springer US, New York, NY.

Joblib Development Team (2008–2021). Joblib: running Python functions as pipeline jobs. https://joblib.readthedocs.io/ (accessed 24.11.2021).

Krayem, A., Yeretzian, A., Faour, G., and Najem, S. (2021). Machine learning for buildings' characterization and power-law recovery of urban metrics. *PLoS ONE*, 16(1): e0246096.

Lal, T. N., Chapelle, O., Weston, J., and Elisseeff, A. (2006). *Feature Extraction: Foundations and Applications*, chapter 5: Embedded Methods, pages 136–165. Springer.

Lánský, I. (2020). Height inference for all USA building footprints in the absence of height data. Master's thesis, Delft University of Technology.

Louppe, G. (2015). *Understanding Random Forests: From Theory to Practice*. PhD thesis, University of Liège.

Lwin, K. and Murayama, Y. (2009). A GIS Approach to Estimation of Building Population for Microspatial Analysis. *T. GIS*, 13:401–414.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Milojevic-Dupont, N., Hans, N., Kaack, L., Zumwald, M., Andrieux, F., de Barros Soares, D., Lohrey, Steffen andPichler, P.-P., and Creutzig, F. (2020). Learning from urban form to predict building heights. *PLOS ONE*, 15(12):1–22.

Ministry of the Interior and Kingdom Relations (2012). Bouwbesluit Online 2012. https://rijksoverheid.bouwbesluit.com/Inhoud/docs/wet/bb2003_nvt/artikelsgewijs/hfd4/afd4-5/art4-24 (accessed 22.12.2021).

Mulder, D. T. (2015). Automatic repair of geometrically invalid 3D City Building models using a voxel-based repair method. Master's thesis, Delft University of Technology.

Müller, A. C. and Guido, S. (2016). *Introduction to Machine Learning with Python*. O'Reilly Media, Inc.

Nouvel, R., Zirak, M., Dastageeri, H., Coors, V., and Eicker, U. (2014). Urban Energy Analysis based on 3D City Model for National Scale Applications. In *Proceedings of the Fifth German-Austrian IBPSA Conference (BauSIM 2014)*.

OGC (2012). OGC City Geography Markup Language (CityGML) Encoding Standard 2.0.0. Technical report, Open Geospatial Consortium.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rokach, L. (2010). *Data Mining and Knowledge Discovery Handbook*, chapter 14: A survey of Clustering Algorithms, pages 269–298. Springer US, Boston, MA.

Scikit-learn (2007–2021a). Cross-validation: evaluating estimator performance. Scikit-learn 1.0.1 Documentation. https://scikit-learn.org/stable/modules/cross_validation.html (accessed 24.11.2021).

Scikit-learn (2007–2021d). Support Vector Machines. Scikit-learn 1.0.1 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html (accessed: 27.10.2021).

Scikit-learn (2007-2021b). Permutation feature importance. Scikit-learn 1.0.1 Documentation. https://scikit-learn.org/stable/modules/permutation_importance.html#id2 (accessed 28.11.2021).

Scikit-learn (2007-2021c). Permutation Importance vs Random Forest Feature Importance (MDI). Scikit-learn 1.0.1 Documentation. https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html (accessed 28.11.2021).

Shiravi, S., Zhong, M., Beykaei, S. A., Hunt, J. D., and Abraham, J. E. (2015). An assessment of the utility of LiDAR data in extracting base-year floorspace and a comparison with the census-based approach. *Environment and Planning B: Planning and Design*, 42(4):708–729.

Smola, A. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14:199–222.

Strobl, C. (2008). *Encyclopedia of GIS*, chapter PostGIS, pages 891–898. Springer US, Boston, MA.

Sullivan, C. B. and Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450.

The PostgreSQL Global Development Group (1996–2021). Chapter 30. Reliability and the Write-AheadLog. PostgreSQL 11.7 Documentation. https://www.postgresql.org/docs/11/wal-intro.html (accessed: 24.11.2021).

Toloşi, L. and Lengauer, T. (2011). Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994.

Ward, J. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.